Macroeconomic Effects of Production Networks: A Deep Learning Approach*

Jesús Fernández-Villaverde
University of Pennsylvania

Christian Matthes
University of Notre Dame

Hikaru Saijo
University of California Santa Cruz

Felipe Schwartzman
Federal Reserve Bank of Richmond

March 2026

Abstract

We develop a scalable deep learning framework to globally solve high-dimensional dynamic general equilibrium models with production networks. Exploiting the symmetry of the economic environment, our algorithm overcomes the curse of dimensionality, and we apply it to a calibrated multi-sector model of the U.S. economy. Global nonlinearities dampen ordinary aggregate fluctuations relative to perturbation methods, yet they slow the recovery from large shocks. The mechanism is a bottleneck in investment hubs, the few sectors that produce most investment goods: their nonlinear resource constraints block the reallocation of resources and reproduce the slow recovery of output and investment after the Great Recession. Perturbation solutions miss this, overstating volatility and predicting a counterfactual rebound. They fail because the capital Euler equation contains singularities that bound the radius of convergence, and the network compounds the problem: a shock of empirically relevant size to one sector propagates through linkages and drives several sectors toward their singularities. We propose a network statistic that predicts where perturbation methods break down.

JEL Codes: C45, C63, C67, E22, E32.

Keywords: Production networks, deep learning, symmetry, nonlinearities.

*jesusfv@econ.upenn.edu, cmatthes@nd.edu, hsaijo@ucsc.edu, and Felipe.Schwartzman@rich.frb.org. We thank the participants at the ASSA 2026 and the Richmond Fed for helpful comments. We are especially grateful to Mahdi Ebrahimi Kahou for useful discussions at various stages of this project.

1 Introduction

The Great Recession was the most severe economic downturn since the 1930s, yet its depth alone does not set it apart. The puzzle is the slow recovery that followed. Output and investment fell sharply in 2008 and 2009 and, unlike in earlier recessions, stayed below their pre-recession trends for nearly a decade. Standard macroeconomic models struggle with this persistence. Once the negative shocks fade, the economy should rebound as firms rebuild depleted capital. Why didn't it?

We argue that the answer lies in global nonlinearities in production networks that standard solution methods neglect. A few sectors (construction, machinery, and motor vehicles) produce most of the economy's investment goods (vom Lehn and Winberry, 2022). When large shocks hit these investment hubs, nonlinear resource constraints create bottlenecks that prevent a quick reallocation of resources toward investment goods. Perturbation methods approximate these constraints locally, miss the bottlenecks, and predict a large, counterfactual rebound.

To capture these nonlinearities, we develop a scalable deep learning framework that globally solves high-dimensional dynamic general equilibrium models with production and investment networks. The algorithm overcomes the curse of dimensionality by exploiting the symmetry of the economic environment. The deep learning solution satisfies the model's equilibrium conditions accurately; the perturbation methods do not.

Our central result is a paradox. The nonlinearities that make ordinary aggregate fluctuations smaller are the ones that make recovery from a large shock slower. In normal times, the nonlinear resource constraints dampen the volatility of output and investment relative to perturbation methods. After a large shock they bind, holding back the reallocation toward investment goods and reproducing the slow recovery seen after the Great Recession. Perturbation methods miss the constraints, so they overstate volatility in normal times and predict a counterfactual boom in the recovery.

Perturbation methods fail for a concrete reason. The Euler equation for capital contains a singularity: at some investment prices the user cost of capital falls to zero and capital demand diverges. A Taylor series has a finite radius of convergence, fixed by the nearest singularity, so no order of perturbation can reach the solution beyond it. In a network these singularities compound, since a shock of empirically relevant magnitude to one sector propagates through linkages and pushes several sectors toward theirs. We summarize the vulnerability in one statistic, the singularity exposure index, computed from the calibration parameters, and show that it is the dominant predictor of the gap between the deep learning and perturbation solutions.

Exploiting symmetry. The main obstacle in analyzing production networks is the size of the state space. Take an n-sector real business cycle model in which each sector receives its own TFP shock. The state has n capital stocks and n productivity levels, which makes standard global methods such as value function iteration infeasible once n is large. The literature has relied on local approximations around a non-stochastic steady state.

We build on Ebrahimi Kahou et al. (2021) and exploit the symmetry of the competitive equilibrium. Conventional global methods treat the aggregate state as an ordered vector $S = (s_1, s_2, \ldots, s_n)$, with each s_i holding sector i's capital and productivity. The ordering is what makes the problem explode: the policy function must record how sector i responds to sector j's state as opposed to sector k's, and the number of such pairwise interactions grows with n. But a sector's effect on the aggregate economy is exchangeable. If sectors j and k swapped their capital and productivity, equilibrium prices would not change. We can still allow for sectoral heterogeneity by adding features such as a sector's position in the network to its state. This lets us treat the aggregate state as a set $S = \{s_1, \ldots, s_n\}$ rather than an ordered vector, which cuts the dimensionality of the problem.

A deep learning implementation. We approximate the equilibrium value and policy functions with deep neural networks. Neural networks are universal approximators, able to represent any continuous function to arbitrary accuracy (Hornik, 1991), which suits the unknown, possibly nonlinear equilibrium policies we face and imposes no parametric assumptions. To carry the symmetry-based reduction into the network, we use the "Deep Sets" architecture (Zaheer et al., 2017). It enforces exchangeability across sectors and keeps the number of network parameters independent of n, so the method scales. We fit the weights by minimizing the squared residuals of the equilibrium conditions along trajectories the network itself simulates. Generating the training data this way concentrates approximation power on the economy's ergodic set, the region of the state space where the economy actually spends its time.

Aggregate dynamics and sectoral propagation. We compare the deep learning solution with first-order and third-order perturbation methods. Global nonlinearities work as a stabilizing force: the deep learning solution produces lower standard deviations of aggregate output and investment than either perturbation solution. The nonlinearity also shifts the economy's ergodic set away from the non-stochastic steady state. Behind both differences is the same fact, that the perturbation solutions violate the sectoral resource constraints by wide margins.

Nonlinearities also change which sectors matter for the business cycle. Under both perturbation methods, the four investment hubs of vom Lehn and Winberry (2022) (construction, machinery, motor vehicles, and professional and technical services) drive aggregate fluctua-

tions. The deep learning solution dampens the elasticities of three of them: construction, machinery, and motor vehicles. Professional and technical services is the exception, and under deep learning it becomes the main source of aggregate labor and investment fluctuations.

The Great Recession and the slow recovery. Because the methods disagree most on investment hubs, their predictions split during episodes with large investment declines, the Great Recession chief among them. All three solutions reproduce the crash of 2008 and 2009, then part ways in the recovery. The perturbation solutions predict a boom, with output and investment climbing well above pre-recession levels. The deep learning solution instead reproduces the stagnation in the data. As sectoral TFP recovers, prices in investment hubs fall, and the perturbation solutions treat this as a reason to shift resources into investment goods. The deep learning solution enforces the nonlinear resource constraints, which block that reallocation and create bottlenecks: depleted capital lowers the marginal product of labor, which lowers labor demand and caps investment goods production, so capital never recovers. This feedback loop is the slow recovery.

Perturbation solutions and singularity: a network-based statistic. What lies behind the poor performance of the perturbation methods? The Euler equation for capital contains a singularity: at some investment prices the user cost of capital falls to zero and capital demand diverges. A Taylor series has a finite radius of convergence, fixed by the nearest singularity, so no order of perturbation can approximate the solution past that radius. The location of sector i's singularity depends on its depreciation rate δ_i, and in our calibration shocks of empirically relevant magnitude push investment prices close to it.

In a multi-sector model the damage depends on how shocks travel through the investment network. We build the singularity exposure index from three ingredients: how strongly a shock to sector i moves investment prices across the network, how close each affected sector sits to its singularity, and how volatile sector i's TFP is. The index follows from the calibration parameters, with no need to solve the model. It is the dominant predictor of the gap in implied aggregate volatility between the deep learning and perturbation solutions, both in simple correlations and as the only variable that stays significant in a regression controlling for sectoral characteristics such as Domar weights and depreciation rates.

Related literature. This paper connects to three literatures. First, a growing body of work applies machine learning to solve macroeconomic models, including Maliar et al. (2021), Azinovic et al. (2022), Fernández-Villaverde et al. (2023), and Kase et al. (2025); Fernández-Villaverde et al. (2024) surveys it. We use the insight of Ebrahimi Kahou et al. (2021) to build a scalable deep learning algorithm for a quantitative production network model.

Second, we contribute to the literature on the macroeconomic consequences of input-

output linkages, including Foerster et al. (2011), Acemoglu et al. (2012), and vom Lehn and Winberry (2022). Much of this work is set in linear environments. Baqaee and Farhi (2019) characterize the second-order general equilibrium effect of sectoral shocks and stress departures from Cobb-Douglas. Their setting is static; we show that nonlinearities arise even under Cobb-Douglas, through the dynamics of capital accumulation.

Third, our paper is close to Carvalho et al. (2025), who use the "deep equilibrium nets" method of Azinovic et al. (2022) to show that static complementarities drive precautionary capital pre-allocation and aggregate nonlinearities. Their effects rely on non-unitary elasticities; ours come from the interaction of capital accumulation and sectoral resource constraints, even under Cobb-Douglas. Our method also complements their direct global solution, using symmetry to handle the high-dimensional state space.

2 The Model

We study a real business cycle model with intersectoral linkages (Acemoglu et al., 2012) and an investment network as in vom Lehn and Winberry (2022). The economy has a representative household and n sectors. Each sector holds a continuum of perfectly competitive firms.

The household owns the firms, consumes, and supplies labor. Lifetime utility is

$$U = \mathbb{E}_0 \sum_{t=0}^{\infty} \beta^t \left[\frac{C_t^{1-\sigma}}{1-\sigma} - \frac{1}{1+\eta} \left(\sum_{i=1}^{n} h_{i,t} \right)^{1+\eta} \right],$$

where C_t is total consumption at time t, $h_{i,t}$ is labor supplied to sector i, σ is risk aversion, β is the discount factor, and η is the inverse Frisch elasticity.

The household aggregates consumption with a Cobb-Douglas aggregator:

$$C_t = \prod_{i=1}^{n} c_{i,t}^{\omega_i}, \tag{1}$$

where ω_i is the preference share of sector i goods in total consumption and $\sum_{i=1}^{n} \omega_i = 1$. Labor is traded in a common market at real wage w_t.

Each firm in sector $i = 1, \ldots, n$ produces

$$y_{i,t} = e^{z_{i,t}} k_{i,t}^{\alpha_i^k} h_{i,t}^{\alpha_i^h} \prod_{j=1}^{n} m_{ij,t}^{a_{ij}}, \tag{2}$$

where $\alpha_i^k + \alpha_i^h + \sum_{j=1}^n a_{ij} = 1$, so production has constant returns to scale. The matrix A summarizes the input-output structure: its (i,j) element a_{ij} is sector i's use of sector j goods in production.

Productivity $z_{i,t}$ is sector-specific and follows an AR(1) process,

$$z_{i,t} = \rho_i^z z_{i,t-1} + \varepsilon_{i,t}^z,$$

where $\boldsymbol{\varepsilon}_t^z \equiv [\varepsilon_{1,t}^z, \ldots, \varepsilon_{n,t}^z]'$ is i.i.d. multivariate Normal with mean zero and covariance matrix Σ^z. Here $k_{i,t}$ is capital and $m_{ij,t}$ is the amount of good j used to produce good i.

Capital evolves as

$$k_{i,t+1} = (1 - \delta_i)k_{i,t} + x_{i,t}, \tag{3}$$

where δ_i is the depreciation rate and $x_{i,t}$ is investment. Firms build capital by combining sectoral goods:

$$x_{i,t} = \prod_{j=1}^n q_{ij,t}^{b_{ij}}, \tag{4}$$

where $\sum_{j=1}^n b_{ij} = 1$ and $q_{ij,t}$ is the amount of sector j goods used for sector i investment $x_{i,t}$. The matrix B summarizes the investment network: its (i,j) element b_{ij} is sector i's use of sector j goods in investment.

The market-clearing condition for sector i is

$$y_{i,t} = c_{i,t} + \sum_{j=1}^n q_{ji,t} + \sum_{j=1}^n m_{ji,t}. \tag{5}$$

3 Equilibrium

Our goal is to solve for the equilibrium recursive law of motion. We denote the individual states in sector i as $s_i \equiv [k_i \quad z_i]$. We then define the aggregate state S as the $n \times 2$ matrix formed by stacking all individual sector states:

$$S \equiv \begin{bmatrix} s_1 \\ s_2 \\ \vdots \\ s_n \end{bmatrix} = \begin{bmatrix} k_1 & z_1 \\ k_2 & z_2 \\ \vdots & \vdots \\ k_n & z_n \end{bmatrix}$$

Thus, this matrix S contains all $2n$ state variables, organized by sector. We first define a recursive competitive equilibrium for this economy.

Definition 1 *A recursive competitive equilibrium is a collection of value functions for firms* $V_i(s_i, S)$*, a set of allocations* $C(S)$*,* $c_i(s_i, S)$*,* $h_i(s_i, S)$*,* $y_i(s_i, S)$*,* $k_i'(s_i, S)$*,* $x_i(s_i, S)$*,* $m_{ij}(s_i, S)$*,* $q_{ij}(s_i, S)$ *and prices* $p_i(s_i, S)$*,* $w(S)$ *such that, for all states:*

1. *Household Optimization: Given prices, the household's allocations solve its utility maximization problem.*

2. *Firm Optimization: Given prices, each firm* i*'s allocations solve its Bellman equation:*

$$V_i(s_i, S) = \max_{m_{ij}, q_{ij}, h_i} \left\{ C^{-\sigma} \left[p_i y_i - \sum_{j=1}^{n} p_j m_{ij} - \sum_{j=1}^{n} p_j q_{ij} - w h_i \right] + \beta \mathbb{E} \left[V_i(s_i', S') \right] \right\}, \quad (6)$$

subject to the constraints

$$\begin{aligned} y_i &= e^{z_i} k_i^{\alpha_i^k} h_i^{\alpha_i^h} \prod_{j=1}^{n} m_{ij}^{a_{ij}}, \\ k_i' &= (1 - \delta) k_i + x_i, \\ x_i &= \prod_{j=1}^{n} q_{ij}^{b_{ij}}. \end{aligned}$$

3. *Market Clearing: For every sector* i*, the resource constraint is satisfied:*

$$y_i = c_i + \sum_{j=1}^{n} q_{ji} + \sum_{j=1}^{n} m_{ji}.$$

To solve this system, we must characterize the policy functions that depend on the economy's state. The necessity of the full state matrix S is a direct consequence of the model's general equilibrium and production network structure. The optimal decisions of an individual firm i depend on the complete vector of current and expected future prices $\boldsymbol{p} \equiv [p_1, \ldots, p_n]'$, w, and the stochastic discount factor $C^{-\sigma}$. The input-output and investment networks create a deep interdependence; Because the networks connect all sectors, even if indirectly, a shock to a single, remote industry l can propagate through these input-output and investment channels, altering the entire price vector $\boldsymbol{p}$. Therefore, to form expectations, firms in sector i need to track the state of every industry in the economy; the matrix S is the minimal sufficient statistic for this forecasting problem. Thus, the equilibrium functions for any firm i depend on the full state matrix S. This $2n$-dimensional state space makes a solution using traditional methods computationally infeasible due to the curse of dimensionality (Bellman (1957)).

Exploiting symmetry. To overcome the curse of dimensionality, we follow Ebrahimi Kahou et al. (2021) and exploit the symmetry inherent in the recursive competitive equilibrium. Intuitively, the optimal decision for the firm in sector i depends on its own states and the distribution of all other sectors' states, but not on the specific identity of those other sectors. This property is called permutation invariance.

This symmetry holds even in our model with heterogeneous sectors. We handle this by including heterogeneity between sectors, such as their production elasticities, network connections (a_{ij}, b_{ij}), depreciation rates, or preference shares, as fixed states within the vector s_i as suggested by Ebrahimi Kahou et al. (2021). The key requirement is simply that any two sectors, i and j, would behave identically if they had the exact same state vector. For example, if sector i and sector j were to swap their complete state vectors, the aggregate set of states would be identical. Because the set is the same, the market-clearing mechanism would produce the exact same equilibrium prices and aggregate consumption. This means the value function for firm i is invariant to any permutation of the states of the other $n-1$ sectors. This allows us to represent the high-dimensional aggregate state not as an ordered vector, but as a set, which is the key insight in Ebrahimi Kahou et al. (2021) that allows for a dramatic reduction in the problem's dimensionality.

Formally, we capture sectoral heterogeneity by defining the individual state s_i as a $D_s \times 1$ vector that contains all features that define a sector. These features include its time-varying states such as capital k_i and productivity z_i, as well as all of its time-invariant parameters, which we collect as:

$$s_i \equiv [k_i, z_i, \omega_i, \alpha_i^k, \alpha_i^h, \delta_i, \rho_i^z, \sigma_i^z, \zeta(a_{i:}, a_{:i}, b_{i:}, b_{:i})]'.$$

The function $\zeta : \mathbb{R}^{4n} \to \mathbb{R}^{L_\zeta}$ is a feature extractor that compresses the $4n$ network parameters for sector i into a low-dimensional summary vector, where $L_\zeta < 4n$. The inputs to ζ are the vectors $(a_{i:}, a_{:i})$ and $(b_{i:}, b_{:i})$, which are the i-th row and column of the input-output matrix A and the investment network matrix B, respectively. These vectors fully define sector i's linkages as a user and supplier of both intermediate and investment goods. This step has two main advantages. First, it makes the problem more tractable by reducing the number of input features, filtering out potentially redundant or sparse information from the raw $4n$ network vector. Second, it ensures the individual state s_i has a consistent dimension, which allows our model's architecture to be general and independent of the specific number of sectors n we are modeling, ensuring the scalability of our solution method.

We can now drop the i-subscript on sectoral values and policies because we have embedded all time-invariant heterogeneity inside the individual state vector s_i. This step recasts

the problem: instead of solving for n different, sector-specific functions (e.g., $V_i(\cdot)$, $c_i(\cdot)$), we now solve for a single, universal set of functions (e.g., $V(\cdot)$, $c(\cdot)$) that are common to all sectors. For instance, the value for sector i is now $V_i = V(s_i, S)$ and the value for sector j is $V_j = V(s_j, S)$, where the mathematical function $V(\cdot)$ is identical for both. This reformulation is what allows us to fully exploit the permutation invariance. Since all equilibrium functions are now common to all sectors, and thus must be invariant to the ordering of the rows in S, we can treat $S = \{s_i\}_{i=1}^n$ as the *set* of all states, instead of an ordered matrix of all states.

This permutation-invariant structure allows us to use a powerful representation theorem (Proposition 2, Ebrahimi Kahou et al. (2021)). In a nutshell, any permutation-invariant function (such as the value function and the policy functions) can be approximated by a structure of the form:

$$f(s_i, S) \approx \rho\left(s_i, \frac{1}{n}\sum_{i=1}^{n}\phi(s_i)\right), \tag{7}$$

where the functions $\rho : \mathbb{R}^{D_s+L_\phi} \to \mathbb{R}$ and $\phi : \mathbb{R}^{D_s} \to \mathbb{R}^{L_\phi}$ are continuous. This architecture, known in the machine learning literature as a Deep Set (Zaheer et al. (2017)), is composed of three distinct components. First, an encoder function, $\phi(\cdot)$, maps each individual state vector s_i into a latent representation. Second, a permutation-invariant pooling layer aggregates these latent representations into a single vector $\frac{1}{n}\sum\phi(s_i)$. This pooled vector serves as a low-dimensional sufficient statistic for the entire distribution of states. Finally, a decoder function, $\rho(\cdot)$, combines the agent's own individual state s_i with this pooled aggregate vector to produce the final output.

4 A Deep Learning Implementation

We are interested in obtaining a global solution that captures rich nonlinearities arising from the intersectoral linkages. The representation in (7) recasts this problem into that of finding a set of unknown functions common to all sectors. We approximate these unknown functions using deep neural networks. Since deep neural networks are proven universal approximators, theoretically capable of approximating any continuous function to an arbitrary degree of accuracy (Cybenko (1989); Hornik (1991)), the flexibility makes them suited for capturing nonlinear functional forms that define the equilibrium policy and value functions. We therefore define $\mathcal{F}(\theta)$ as the deep learning architecture that parameterizes these functions, where θ is the vector of weights and biases in the network.

Overview. We approximate several equilibrium policy and value functions simultaneously by noting that the representation theorem applies component-wise. We define ρ with

a three-dimensional output to capture the vector $[\hat{c}, \hat{x}, \hat{V}]'$:

$$\begin{bmatrix} \hat{c}\,(s_i, S) \\ \hat{x}\,(s_i, S) \\ \hat{V}\,(s_i, S) \end{bmatrix} \approx \rho \left(s_i, \frac{1}{n} \sum_{i=1}^{n} \phi(s_i); \theta \right),$$

where hat denotes log-deviations (for the value function, deviations) from the non-stochastic steady states. We found through experimentations working with deviations rather than levels facilitates training. Other equilibrium allocations can be computed exactly from the equilibrium conditions, given the approximated functions (details in the Appendix).

We estimate the network weights θ by minimizing the loss functions composed of the sum of squared residuals from the model's equilibrium conditions, evaluated at the states visited by our training simulations. We use a stochastic gradient descent method to perform this minimization. An important difference from traditional solution methods, such as value function iteration, is that we do not rely on a fixed grid; instead, we periodically re-simulate the economy using the current policy approximations to generate new training points. This iterative procedure ensures the neural network focuses its approximation power on the economy's ergodic set, learning the global solution where the economy actually resides rather than in irrelevant regions of the high-dimensional state space (Azinovic et al. (2022), Maliar et al. (2021)).

To implement the optimization, we partition the training dataset of length T into mini-batches of size M. We define an *epoch* as one complete pass through the entire dataset, which corresponds to performing T/M gradient updates. The algorithm iterates through this process for a total of E epochs. All trainings and simulations are conducted on PyTorch.

Network training. We now detail (1) how we select the initial weights (pre-training), (2) the network architecture $\mathcal{F}(\theta)$, (3) a loss-function to evaluate the approximation quality on a given training data, (4) the optimization algorithm over the loss-function, and (5) the training data generation scheme.

1. To facilitate numerical stability and ensure the solution is grounded in a robust benchmark, we initialize the network weights θ by "anchoring" the deep neural network to the first-order perturbation solution. We generate a synthetic dataset of $T_{pre} = 1000$ periods using policy rules derived from a log-linearization around the non-stochastic steady state. We then pre-train the neural network for $E_{pre} = 6000$ epochs to minimize the mean squared error (MSE) between its predictions and the predictions from the log-linear policy. This pre-training ensures that the network begins its search from a reasonable region of the parameter space, allowing the subsequent nonlinear training

to refine the solution specifically where the log-linear approximation fails.

2. The function approximator $\mathcal{F}(\theta)$ is composed of three distinct residual neural networks (ResNets). Unlike standard feedforward networks, we employ skip connections to facilitate the training of deeper architectures. This technique prevents gradient degradation by allowing information to propagate unimpeded through the network layers (He et al. (2016)). The specific architectures are:

   (a) Feature Extractor ($\zeta : \mathbb{R}^{4n} \to \mathbb{R}^{L_\zeta}$): This network compresses the raw network matrices (A and B) into a latent vector of dimension $L_\zeta = 8$. It consists of $K = 7$ layers, each with 500 nodes.

   (b) Encoder ($\phi : \mathbb{R}^{D_s} \to \mathbb{R}^{L_\phi}$): This network maps the sector-specific state vector s_i into a latent representation of dimension $L_\phi = 20$. It consists of $K = 7$ layers, each with 500 nodes.

   (c) Decoder ($\rho : \mathbb{R}^{D_s+L_\phi} \to \mathbb{R}^3$): This network approximates the equilibrium functions. It takes as input the individual state s_i and the L_ϕ-dimensional mean of the latent representations from all sectors. It outputs the vector $[\hat{c}, \hat{x}, \hat{V}]'$. It consists of $K = 7$ layers, each with 1500 nodes.

3. For a batch of M simulated states, the loss $\mathcal{L}(\theta)$ is the weighted sum of four components. We use tilde to denote approximated variables and for notational simplicity suppress individual and aggregate states when there is no confusion. Then

$$\mathcal{L}(\theta) = \frac{1}{Mn} \sum_{m=1}^{M} \left[\sum_{i=1}^{n} \nu \varepsilon_{i,m}^{rewards} + \sum_{i=1}^{n} \left(\varepsilon_{i,m}^{value}\right)^2 + \sum_{i=1}^{n} \left(\varepsilon_{i,m}^{resource}\right)^2 + \sum_{i=1}^{n} \left(\varepsilon_{i,m}^{Euler}\right)^2 \right], \quad (8)$$

where

$$\varepsilon_i^{rewards} = -\left(\widetilde{C}^{-\sigma} \left[\widetilde{p}_i \widetilde{y}_i - \sum_{j=1}^{n} \widetilde{p}_i \widetilde{m}_{ij} - \sum_{j=1}^{n} \widetilde{p}_j \widetilde{q}_{ij} - \widetilde{w} \widetilde{h}_i \right] + \beta \mathbb{E}\left[\widetilde{V}\left(s_i', S'\right) \right] \right)$$

$$\varepsilon_i^{value} = 1 - \frac{\widetilde{C}^{-\sigma} \left[\widetilde{p}_i \widetilde{y}_i - \sum_{j=1}^{n} \widetilde{p}_i \widetilde{m}_{ij} - \sum_{j=1}^{n} \widetilde{p}_j \widetilde{q}_{ij} - \widetilde{w} \widetilde{h}_i \right] + \beta \mathbb{E}\left[\widetilde{V}\left(s_i', S'\right) \right]}{\widetilde{V}\left(s_i, S\right)}$$

$$\varepsilon_i^{resource} = 1 - \frac{\widetilde{c}_i + \sum_{j=1}^{n} \widetilde{m}_{ji} + \sum_{j=1}^{n} \widetilde{q}_{ji}}{\widetilde{y}_i}$$

$$\varepsilon_i^{Euler} = 1 - \frac{\beta \mathbb{E}\left\{ \widetilde{C}'^{-\sigma} \widetilde{p}_i' \alpha_i^k \frac{\widetilde{y}_i'}{\widetilde{k}_i} + \widetilde{\mu}_i' (1-\delta) \right\}}{\widetilde{\mu}_i}$$

for $i = 1, \ldots, n$. Here $\varepsilon_i^{rewards}$ concerns the maximization of the rewards (the right-hand-side of the Bellman equation (6)) and hence the negative sign, and ε_i^{value} concerns the definition of the value function $V(s_i, S)$ in (6). We set the scaling ν of $\varepsilon_i^{rewards}$ so as to ensure that this term is roughly of the same order of magnitude as the other residuals. Since $\varepsilon_i^{rewards}$ is measured in levels while the other errors are normalized in deviations, this scaling prevents the gradient descent from being dominated by the scale of the value function. $\varepsilon_i^{resource}$ are the residuals from the sectoral resource constraint and sectoral consumption demand, respectively. Finally, ε_i^{Euler} is the residual from the Euler equation, where $\widetilde{\mu}_i$ is the approximated Lagrangian multiplier for the sector i capital accumulation equation. While the Euler equation residual is not strictly necessary for finding the solution given that we include $\varepsilon_i^{rewards}$ and ε_i^{value}, we found that including it often improves the accuracy.

4. We minimize the loss using AdamW.[1] We use a batch size of $M = 100$ and a Cosine Annealing scheduler, where we initialize the learning rate at 10^{-5} and gradually reduce it to 10^{-6} over $E = 3000$ epochs. This schedule allows the network to explore the loss landscape broadly in early phases and converge to a sharp minimum in later phases. To ensure numerical stability, we apply gradient clipping with a maximum norm of 100.

5. We employ an adaptive simulation scheme. Every 20 epochs, we regenerate the training dataset of $T = 1000$ periods by simulating new trajectories for capital using the network's current policy approximation, combined with new random draws for productivity. We initialize these new simulations by setting the initial states to the mean of the current training set. To maintain stability and prevent catastrophic forgetting (McCloskey and Cohen (1989)), where the network overfits to the new data and forgets previous policies, we utilize an "experience replay" buffer. Each new training data consists of a random mix of 75% current data points and 25% newly simulated points.

We summarize our steps in the pseudocode below.

Discussion. We discuss three points regarding the deep learning implementation: (1) how our network training specification helps select the stable solution, (2) how our selection of the training dataset size is motivated by the double descent phenomenon, and (3) the computation of conditional expectations.

1. In the context of deep learning, Ebrahimi Kahou et al. (2024) argue that the inductive bias of neural networks toward simple, minimum-norm solutions can naturally select

[1] AdamW is a stochastic optimization algorithm that decouples weight decay from adaptive gradient updates, offering better regularization than standard Adam.

Algorithm: Network training

Step 1. Pre-training
Initialize the network weights θ by minimizing the MSE between the network and log-linear policy predictions.

Step 2. Data generation
Periodically regenerate the training data by simulating trajectories using the current policy approximation $\rho(\cdot;\theta), \phi(\cdot;\theta), \zeta(\cdot;\theta)$.

Step 3. Optimization loop
For each batch in the training data:
(a) Compute the equilibrium residuals and construct the loss $\mathcal{L}(\theta)$ in (8).
(b) Compute gradients $\nabla_\theta \mathcal{L}(\theta)$.
(c) Update weights θ using the AdamW optimizer.

Step 4. Termination
Repeat Steps 2 and 3 until a pre-specified maximum number of epochs is reached.

stable trajectories without explicit boundary conditions, provided that the approximated functions are chosen such that the stable solution exhibits smaller functional norms than the explosive paths. Consistent with this insight, we found through experimentation that in our high-dimensional setting, a loss function based solely on Euler equation errors ε^{Euler} was insufficient to rule out explosive dynamics. This suggests that, in the absence of a global value constraint, the explosive trajectories appeared "simpler" to the network (in a functional norm sense) than the true stable path. However, by augmenting the loss function with the Bellman equation residuals ($\varepsilon^{rewards}$ and ε^{value}), we effectively anchor the approximated functions to the stable solution, ruling out explosive paths. Since the Bellman operator is a contraction mapping with a unique fixed point, enforcing this condition eliminates the multiplicity of explosive solutions permissible under the Euler equation alone.

2. Our selection of the training dataset size T is motivated by the double descent phenomenon described in Belkin et al. (2019). Classical statistical theory suggests that models with a number of parameters far exceeding the sample size suffer from overfitting and poor generalization. However, deep neural networks frequently exhibit a second regime of performance improvement: once model capacity surpasses the interpolation threshold (the number of parameters is large enough to exactly fit the data), generalization error often declines as the ratio of parameters to data points increases. Consequently, we choose a sample size T that balances two objectives. First, the

dataset must be sufficiently large to capture the diverse dynamics and ergodic properties of the high-dimensional state space. Second, the sample size must remain small enough relative to the network's capacity to ensure the estimator operates within the over-parameterized regime. This allows us to leverage the implicit bias provided by the stochastic gradient descent algorithm, mitigating overfitting.

3. To evaluate the conditional expectations in the Bellman and Euler equations, we employ a hybrid integration scheme that balances precision with computational efficiency. For the agent's own idiosyncratic state s_i' (specifically the productivity shock z_i'), we use Gauss-Hermite quadrature. However, computing the expectation over the aggregate state S', which depends on the vector of shocks to all n sectors, is intractable with quadrature. Instead, we rely on the concentration of measure property (Ebrahimi Kahou et al. (2021)): since the aggregate sufficient statistic $\frac{1}{n}\sum \phi(s_i')$ averages over many independent shocks, it concentrates sharply around its mean. Thus we approximate this component using a Monte Carlo estimator with a small random number of draws with $R_{mc} = 4$; that is, we simulate just four independent realizations of the entire vector of n sectoral shocks to estimate the expected aggregate state. While asymptotically consistent, using such a small sample size also provides a practical benefit during training: the resulting stochastic noise prevents the optimization algorithm from settling into sharp local minima, acting as a form of implicit regularization.

5 Quantitative Analysis

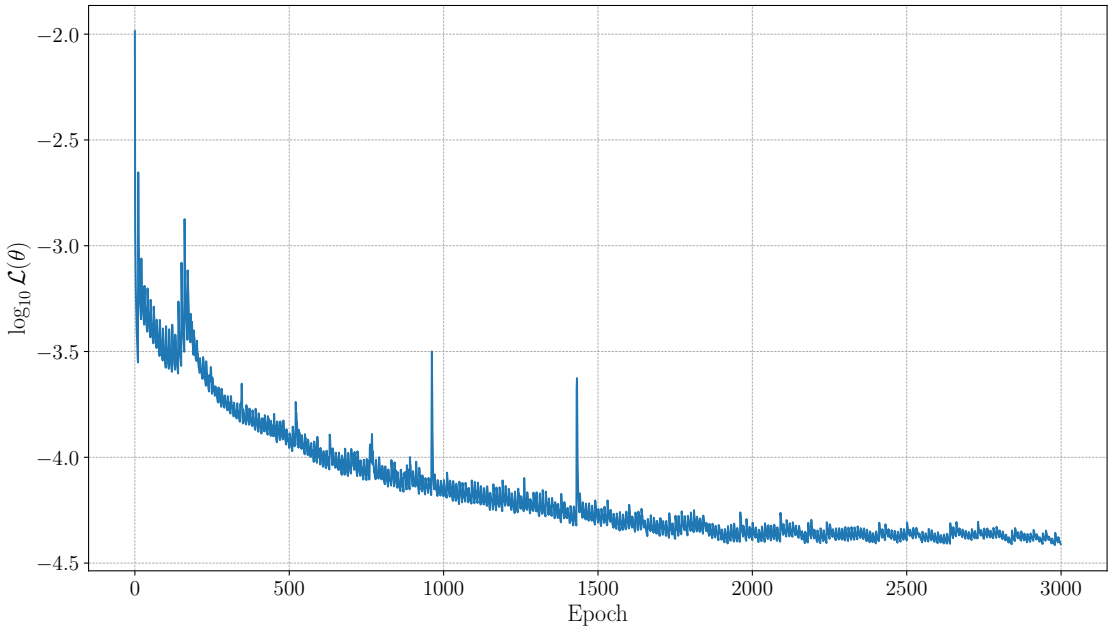
We calibrate the model to an annual frequency for $n = 37$ sectors.[2] We set the discount factor to $\beta = 0.96$. The risk aversion parameter is set to $\sigma = 1$, which implies log utility. The inverse Frisch elasticity is set to $\eta = 0$, which follows the indivisible labor model of Hansen (1985). For the input-output matrix A, the investment network matrix B, preference shares $\{\omega_i\}_{i=1}^n$, capital shares $\{\alpha_i^k\}_{i=1}^n$, labor shares $\{\alpha_i^h\}_{i=1}^n$, depreciation rates $\{\delta_i\}_{i=1}^n$, and the TFP parameters $\{\rho_i^z\}_{i=1}^n$ and Σ^z, we set to values provided by vom Lehn and Winberry (2022).

5.1 Results

We report the performance of our training and present several results. Figure 1 plots the path of the loss function $\mathcal{L}(\theta)$ over the training run, showing a consistent downward trend. The occasional spikes, particularly evident in the early phases, correspond to the periodic regeneration of the training dataset via adaptive simulation. These temporary increases

[2] In the Appendix, we list all 37 sectors.

Figure 1: The path of training loss



in loss reflect the network encountering new regions of the ergodic set where the policy approximation has not yet been refined. This process ensures the network expands its domain of accuracy to cover the true ergodic set.

Table 1: Test errors

	MAE	MSE	Max	P10	P50	P90
Panel A: Deep learning solution						
Euler equation	9.7e-3	2.0e-4	7.2e-2	1.0e-3	6.1e-3	2.4e-2
Resource constraint	9.0e-3	1.8e-4	1.1e-1	1.1e-3	6.0e-3	2.0e-2
Panel B: First-order solution						
Euler equation	1.4e-2	7.8e-4	5.3e-1	1.3e-3	7.1e-3	3.2e-2
Resource constraint	5.9e+2	1.5e+8	8.7e+5	2.2e-4	1.1e-2	3.3e+1
Panel C: Third-order solution						
Euler equation	1.2e-2	1.4e-3	1.1e+0	5.2e-4	3.9e-3	2.2e-2
Resource constraint	inf	inf	inf	9.4e-3	1.2e-1	inf

We assess the accuracy of our deep learning solution by comparing it against first-order (log-linearization) and third-order perturbation solutions. We construct a test set by sampling 200 states from the deep learning ergodic path. Panel A of Table 1 evaluates the deep learning solution on these states. For the perturbation solutions in Panels B and C, we demean the capital stocks from the deep learning path so that the test states are centered at the non-stochastic steady state, and then apply the perturbation decision rules at these demeaned states. This ensures we evaluate the perturbation methods at states with

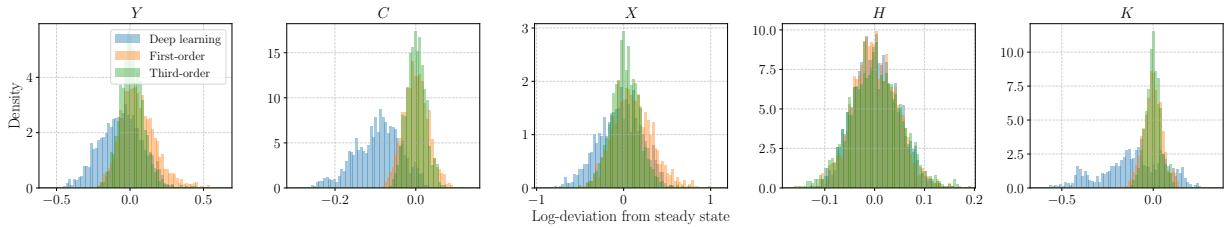
the same stochastic property as the deep learning ergodic set, but centered at their expansion point. For each equilibrium condition, the table reports the mean absolute error (MAE), the mean squared error (MSE), the maximum absolute error (Max), and the 10th, 50th, and 90th percentiles (P10, P50, P90) of the absolute errors across all observations. The results highlight the advantage of the deep learning approach along both dimensions. For the Euler equation, the deep learning solution achieves the lowest MAE of 9.7×10^{-3}, followed by the third-order solution (1.2×10^{-2}) and the first-order solution (1.4×10^{-2}). The distribution of errors reveals an important distinction between the two perturbation methods. The third-order solution outperforms the first-order solution across most of the distribution, with lower P10, P50, P90, and MAE. However, the third-order solution has a higher MSE (1.4×10^{-3}) than the first-order (7.8×10^{-4}), despite being better on average. This is explained by extreme errors in the tail: the maximum Euler equation error for the third-order solution reaches 1.1, roughly twice the first-order maximum of 5.3×10^{-1} and an order of magnitude larger than the deep learning maximum of 7.2×10^{-2}. These outliers, driven by the instability of the higher-order terms when the economy is far from the steady state, dominate the MSE. For the sectoral resource constraint, the perturbation solutions fail dramatically. The first-order solution produces an MAE of 5.9×10^{2}, indicating that the log-linear decision rules, when evaluated at states with empirically relevant volatility, imply sectoral demands that deviate massively from available output. The third-order solution fares even worse, with errors that diverge to infinity as the higher-order terms amplify resource constraint violations. In contrast, the deep learning solution maintains a low MAE of 9.0×10^{-3}, demonstrating its ability to respect the nonlinear resource constraints of the production network.

Table 2: Ergodic moments of aggregate variables

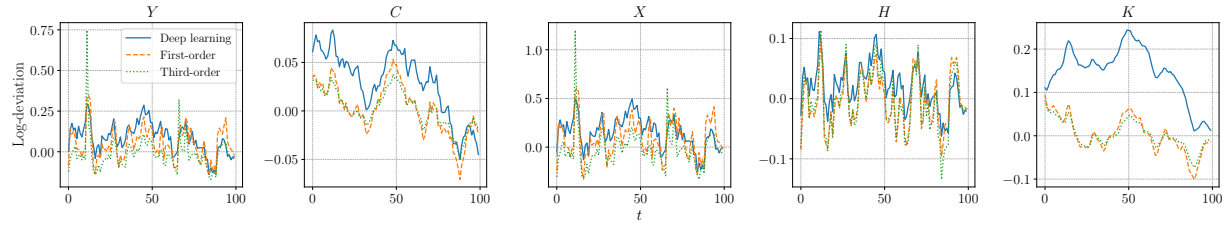
	Y	C	X	H	K
Panel A: Ergodic means					
Deep learning	-0.06	-0.09	-0.04	0.00	-0.11
First-order	0	0	0	0	0
Third-order	0.02	0.00	0.04	0.00	0.00
Panel B: Standard deviations					
Deep learning	5.6	0.8	11.2	2.7	1.0
First-order	9.5	0.9	18.4	3.3	1.1
Third-order	23.7	0.6	31.6	3.8	0.8
Data	2.2	1.8	9.9	2.7	0.8

To visually illustrate the differences between the solution methods, Figure 2 plots the ergodic dynamics of the aggregate variables. For the third-order solution, we use the pruning

Figure 2: Comparison of ergodic distributions and time-series simulations



(i) Ergodic distributions



(ii) Simulated paths

method of Kim et al. (2008) and Andreasen et al. (2018) to simulate stable ergodic paths. Panel (i) displays the histograms of the ergodic distributions, expressed as log-deviations from the non-stochastic steady state. Panel (ii) presents an illustrative 100-period simulation path. To facilitate a direct comparison of volatility and cyclical dynamics, the deep learning and third-order series in Panel (ii) are reported as log-deviations from their respective ergodic means, while the first-order solution paths are expressed as log-deviations from the steady state.

We highlight three features of the deep learning solution. First, the deep learning solution generates ergodic distributions that are shifted away from the non-stochastic steady state. The histograms in Panel (i) show that the deep learning distributions for output and capital are displaced leftward relative to both perturbation solutions. Panel A of Table 2 quantifies these shifts: the ergodic mean of aggregate capital is 0.11 log points below the deterministic steady state, accompanied by lower levels of aggregate output (0.06 log points) and consumption (0.09 log points). In contrast, the third-order solution generates small but positive shifts in ergodic means of output and investment from the steady state. Second, the deep learning solution produces dampened aggregate dynamics relative to the perturbation methods. The time-series paths in Panel (ii) show that the deep learning solution fluctuates with consistently lower amplitude, and Panel B of Table 2 confirms this quantitatively by comparing standard deviations of aggregate variables in log first differences.[3] The standard

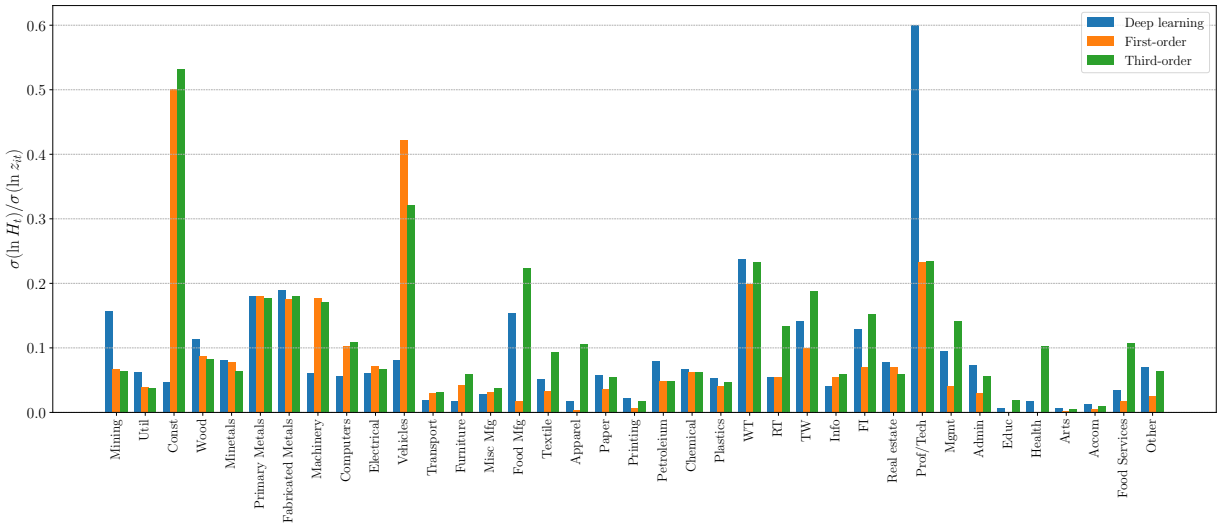
[3]The standard deviations are multiplied by 100 to express them in percentage terms.

deviation of investment is 11.2% for the deep learning solution, compared to 18.4% for the first-order and 31.6% for the third-order approximation. Output volatility exhibits the same ranking. This dampening is consistent with the resource constraint analysis in Table 1: the perturbation methods overestimate the responsiveness of investment to shocks when away from the steady state, whereas the deep learning solution captures the nonlinear resource constraints that naturally attenuate these fluctuations. Comparing with the data, the deep learning solution produces output and investment volatility closer to the empirical values, although all three solutions overstate output volatility. Interestingly, the volatility of labor is comparable across the three solutions, with the deep learning solution being closest to the data.

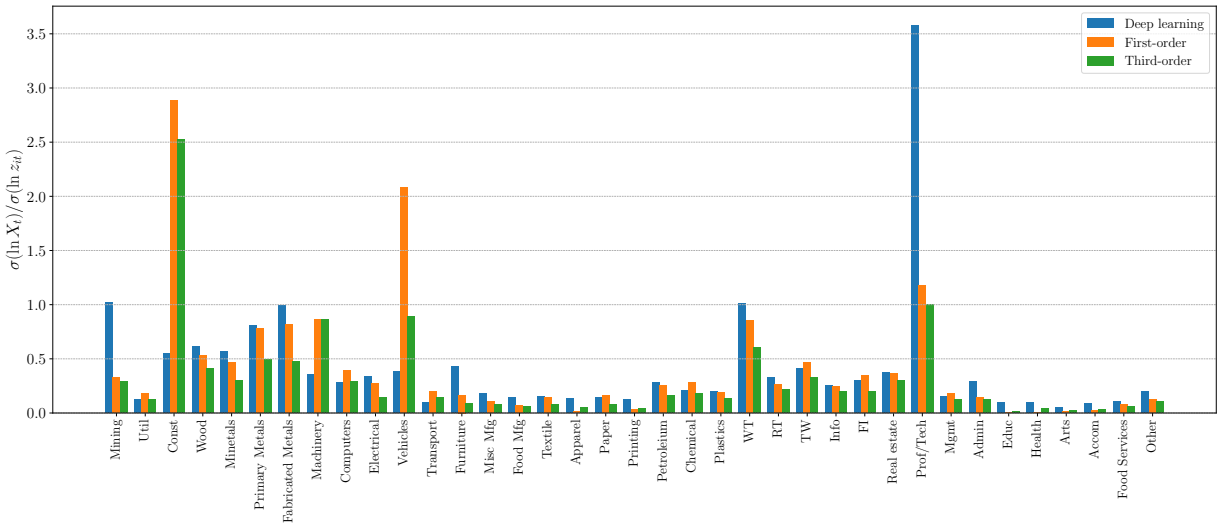
A closer look at the third-order solution reveals a tension. On one hand, the histograms in Panel (i) show that the third-order distributions for output and investment appear tighter than those of the first-order solution, suggesting that the pruned third-order approximation is more stable for most realizations. On the other hand, the time-series paths in Panel (ii) reveal that the third-order solution exhibits occasional microbursts of instability (Andreasen et al., 2018)—brief but extreme deviations in output and investment that inflate its overall volatility well beyond even the first-order level despite its typically moderate fluctuations. Finally, as we show in the Appendix, the distribution of ergodic means is highly heterogeneous across sectors. For instance, while some sectors, such as mining, exhibit ergodic means significantly above their deterministic steady states, other sectors, particularly construction or those in durable goods manufacturing, show means below the steady states.

To identify which sectors drive aggregate fluctuations, we compute numerical elasticities of aggregate labor and investment to sector-specific TFP shocks following the methodology of vom Lehn and Winberry (2022). For each sector i, we simulate the economy where only sector i experiences productivity shocks (scaled by its empirical volatility σ_i^z), while shocks to all other sectors are shut down. We then calculate the elasticity as the ratio of the volatility of aggregate labor and investment to the volatility of the sectoral TFP. This metric thus measures the strength of each sector's influence on aggregate labor and investment through the production network. Figure 3 plots these elasticities. The results reveal an important reallocation of systemic importance under the deep learning solution. The perturbation methods not only overstate aggregate volatility but qualitatively misidentify the sectoral sources of business cycles, overemphasizing construction and durable manufacturing at the expense of service sectors. Under both perturbation methods, the construction and vehicles sectors are by far the dominant drivers of aggregate labor and investment fluctuations, replicating the finding by vom Lehn and Winberry (2022). The deep learning solution, however, drastically dampens the elasticities of these investment hub sectors. Instead, under

Figure 3: Elasticity of aggregate variables to sectoral TFP shocks



a Aggregate Labor



b Aggregate Investment

deep learning the professional and technical services sector emerges as the most important sector for both aggregate labor and investment, with an investment elasticity roughly three times that of the perturbation solutions. The biggest gap between the deep learning and perturbation solutions thus lies precisely in the investment hubs, which produce nearly 70% of total investment (vom Lehn and Winberry, 2022).

5.2 The Great Recession and the Slow Recovery

Since the solution methods disagree most sharply on investment hub sectors, their predictions diverge dramatically during episodes featuring large investment declines, such as the Great Recession and the aftermath. We show that the deep learning solution accounts for the slow recovery of output and investment through bottlenecks in investment hubs, while the perturbation solutions counterfactually predict a large rebound in economic activity.

We start our simulation from 2007, assuming the deep learning and third-order perturbation paths were at their respective ergodic means and the first-order perturbation path were at the steady state. We then use the AR(1) residuals from the sectoral TFP estimates by vom Lehn and Winberry (2022) for 2008–2018 and feed these empirical innovations sequentially into the model. Figure 4 plots, in log-deviations, the paths of aggregate variables alongside the detrended data, normalizing the 2007 level to zero.[4] The bottom-right panel plots aggregate TFP, constructed as Domar-weighted sectoral TFP, where the Domar weight of sector i is $w_i = p_i y_i/\text{GDP}$. Aggregate TFP drops sharply in 2008 and 2009 and remains below trend through 2018, never fully recovering.

All three solutions replicate the initial crash in output, investment, and labor in 2008 and 2009, driven by the drop in TFP. The key difference lies in the recovery. In the data, output and investment remain persistently below their pre-recession trends throughout the period, and labor recovers only slowly. The deep learning solution reproduces these features: output, investment, and labor persistently stay at or below their pre-recession levels, tracking the data, although the model labor path recovers faster than the data. The first-order solution, by contrast, predicts a counterfactual boom: output rises to over 30% above its initial level in 2010, fueled by an investment surge exceeding 60% and a quick labor expansion. Under the third-order solution this extreme overshooting is moderated but does not reproduce the persistent stagnation seen in the data and captured by the deep learning solution.

Table 3 confirms this assessment by comparing the MSEs, computed as $1/T \sum_t (y_t^{model} - y_t^{\text{data}})^2$, for the three solutions. The deep learning solution achieves the best overall fit, with an average MSE an order of magnitude below the perturbation methods. The advantage is especially pronounced for output and investment, where the perturbation methods' counterfactual recoveries drive large deviations from the data. Among the two perturbation methods, the third-order solution improves on the first-order due to the less severe

[4] In the data, output corresponds to GDP, consumption to personal consumption expenditures, and investment to gross private domestic investment, all from the BEA NIPA Table, deflated by the GDP deflator. Labor corresponds to nonfarm business sector hours worked from the BLS. The aggregate capital stock is constructed via the perpetual inventory method using real investment, with the aggregate depreciation rate set to the capital-weighted average of sectoral rates. All series are detrended using fourth-order polynomials, following vom Lehn and Winberry (2022).

overshooting in output and investment.

Figure 4: The Great Recession and the slow recovery: model vs. data

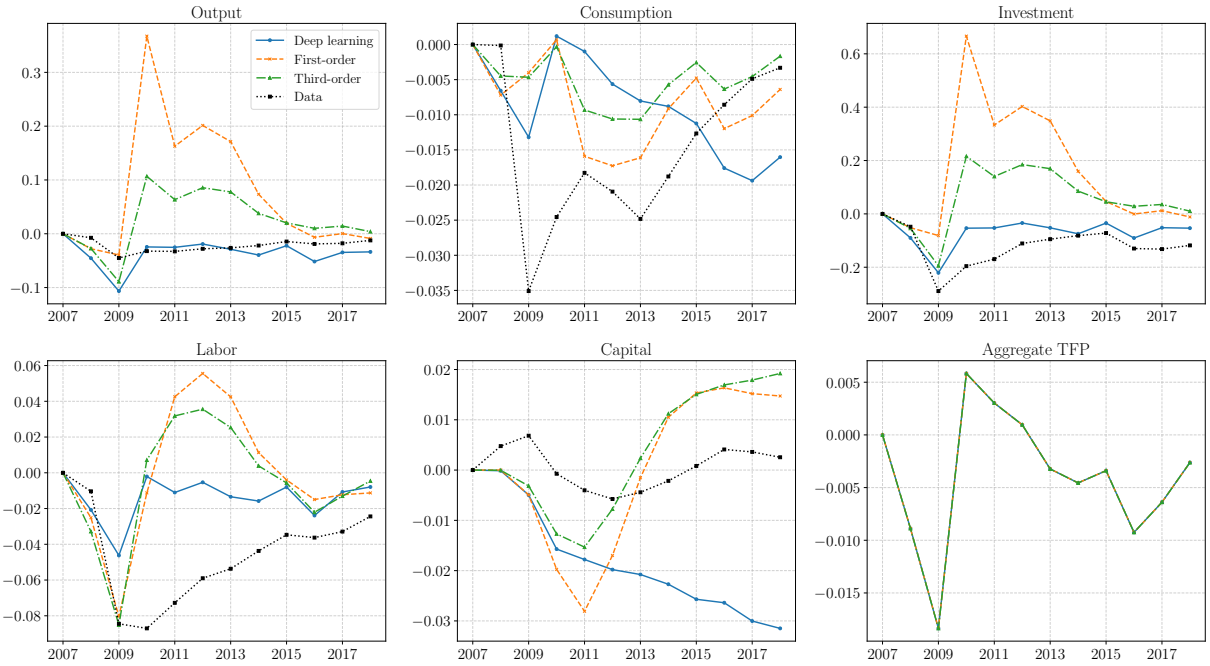


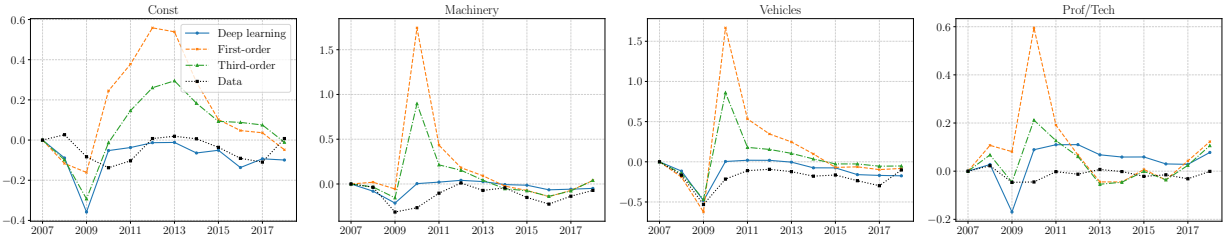
Table 3: Mean squared error: model vs. data

	Y	C	X	H	K	Average
Deep learning	6.3e-4	2.1e-4	5.1e-3	1.6e-3	4.5e-4	1.6e-3
First-order	2.5e-2	1.6e-4	1.4e-1	3.9e-3	1.7e-4	3.3e-2
Third-order	5.1e-3	1.8e-4	4.5e-2	3.3e-3	1.2e-4	1.1e-2

The divergence between the deep learning and perturbation solutions originates in the investment hubs. Figure 5 plots investment goods production, calculated as $\sum_j x_{ji}$ for sector i, in construction, machinery, motor vehicles, and professional and technical services. Under the first-order solution, investment goods production in these sectors rebounds sharply after 2009: construction output rises over 50%, while machinery and motor vehicles each surge by roughly 150%. These booms are counterfactual; the data show that investment goods production in investment hubs remained depressed well into the recovery. The deep learning solution keeps investment production near or below pre-recession levels, consistent with the data. The third-order solution generates intermediate rebounds that are smaller than the first-order but still counterfactually large.

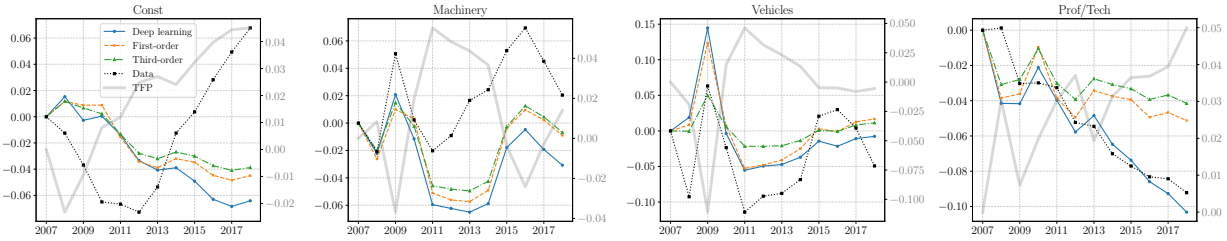
Figure 6 provides evidence for a bottleneck mechanism. Sectoral prices in the investment hubs decline during the recovery, caused by the recovery in sectoral TFP. Under the

Figure 5: Investment production in investment hubs



perturbation solutions, these low prices stimulate a rapid expansion driven by a surge in investment demand. Under the deep learning solution, the same price declines fail to trigger an investment boom. The nonlinear resource constraints that the perturbation methods fail to satisfy, as documented in Table 1, prevent the massive reallocation of resources toward investment goods production that those solutions predict.

Figure 6: Sectoral prices in investment hubs

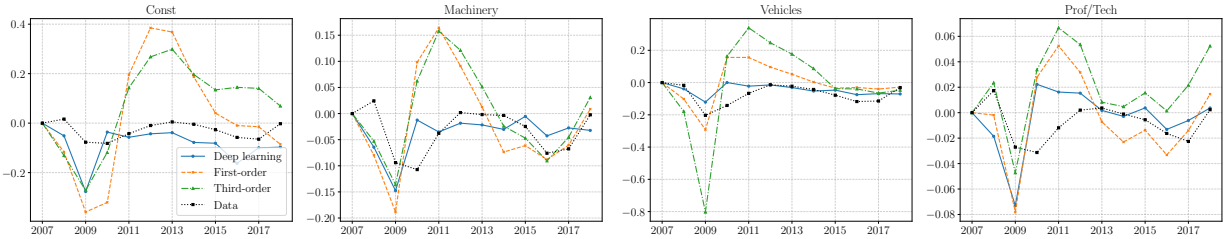


The labor market reveals how the perturbation solutions finance their counterfactual investment booms. Figure 7 shows that under the first-order solution, labor in construction surges over 30% above its initial level, with large expansions also in machinery and motor vehicles. Under the deep learning solution, labor in these sectors remains depressed throughout the recovery. Low capital stocks reduce the marginal product of labor, which depresses labor demand and limits the capacity to expand investment goods production. This creates a feedback loop: depleted capital lowers labor demand, which constrains investment production, which in turn prevents capital from recovering, thus generating the slow recovery observed in the data.

6 Singularities and the Failure of Perturbation Methods: A Network-Based Statistics

What accounts for the failure of perturbation methods? With Cobb-Douglas production function and logarithmic preferences, the static model yields log-linear expressions for allo-

Figure 7: Sectoral labor in investment hubs



cations and prices, so that a first order log-linear approximation would be exact. This is not true, however, in the dynamic model because the production function for capital embedded in the capital accumulation equations (3) is not log-linear when capital does not fully depreciate ($\delta_i < 1$). In effect, even higher order Taylor series approximations will fail due to singularities - points where functions blow up to infinity.

The non-log-linearities are also apparent in the Euler equation. In particular, optimal capital choice in sector i requires

$$p_{i,t}^I = \beta \mathbb{E}_t \left\{ \left(\frac{C_{t+1}}{C_t} \right)^{-\sigma} \left(p_{i,t+1} \alpha_i^k \frac{y_{i,t+1}}{k_{i,t+1}} + (1 - \delta_i) p_{i,t+1}^I \right) \right\} \tag{9}$$

where $p_{i,t}^I \propto \prod_j (p_{j,t})^{b_{ij}}$ is the price of index for the investment good aggregator used in sector i at time t.

If $\delta_i = 1$, the Euler equation is log-linear, and the model solution collapses back to a Long and Plosser (1983) style model that admits an analytic solution. In such models, a constant share of gross-output in each sector flows to consumption, materials or investment goods, $c_{i,t}/y_{i,t}$, $m_{ji,t}/y_{i,t}$ and $q_{ji,t}/y_{i,t}$. With constant shares, the log-linear approximation to the resource constraint is always exact, since it takes the form

$$\varepsilon_i^{resource-log-linear} = 1 - \frac{c_i}{y_i} e^{\widehat{c}_i - \widehat{y}_i} + \sum_{j=1}^{n} \frac{m_{ji}}{y_i} e^{\widehat{m}_{ji,t} - \widehat{y}_{i,t}} - \sum_{j=1}^{n} \frac{q_{ji}}{y_i} e^{\widehat{q}_{ji,t} - \widehat{y}_{i,t}}$$

and all terms in the exponentials remain identical to zero. With $\delta_i < 1$, however, the fraction of output in any given sector dedicated to investment will be variable. In the log-linear approximation the share will be proportional to $e^{\widehat{q}_{ij,t} - \widehat{y}_{i,t}}$. Thus, potential linearization errors in the share get exponentially amplified, leading to potentially large resource constraint violations.

The approximation errors do not disappear with higher order approximations because with $\delta_i < 1$ there are values of endogenous variables for which capital choices are not defined and in the vicinity of which it diverges to infinity. Those become visible once we rearrange

equation (9) to obtain the capital demand in sector i:

$$k_{i,t+1} = \frac{\beta \mathbb{E}_t \left[\left(\frac{C_{t+1}}{C_t} \right)^{-\sigma} p_{i,t+1} \, \alpha_i^k \, y_{i,t+1} \right]}{u_{i,t}},$$

where $u_{i,t}$ is the user cost of capital in sector i, time t:

$$u_{i,t} = p_{i,t}^I - (1 - \delta_i) \beta \, \mathbb{E}_t \left[\left(\frac{C_{t+1}}{C_t} \right)^{-\sigma} p_{i,t+1}^I \right]$$

Thus, capital in sector i, period $t+1$ is only well defined if the user cost is positive, that is, $u_{i,t} \geq 0$. More generally, when $u_{i,t} \to 0$, capital demand $k_{i,t+1} \to \infty$. Intuitively, as $u_{i,t} \to 0$ the discounted resale value of undepreciated capital tomorrow approaches the purchase price today, i.e., capital becomes a near-free asset, so agents want to buy unbounded amounts of it. The reason this introduces approximation errors is that Taylor series have a finite radius of convergence R set by the nearest singularity. Thus if p_s is the point of singularity closest to the point of approximation p_0, $R = |p_s - p_0|$ and a Taylor series converges do the true function as the number of polynomial terms increases for $|p - p_0| < R$. For $|p - p_0| > R$, however, the series diverges. In other words, adding more terms (higher-order perturbation) improves accuracy *within* R but cannot extend R.

We can quantify how close a sector is to its singularity by perturbing the current investment price while keeping the stochastic discount factor, the expected marginal product of capital, and future investment prices fixed at their steady-state values. Under those assumptions,

$$\frac{k_{i,t+1}}{k_i} = \frac{u_i}{u_i + p_i^I (e^{\varepsilon_t} - 1)}$$

where ε captures the perturbation to the current investment price. There is a singularity when the denominator equals zero, that is, for

$$\varepsilon_t^* = \log\big(\beta(1 - \delta_i)\big).$$

For typical calibration ($\beta = 0.96$, $\delta = 0.10$):

$$\varepsilon_t^* = \log(0.864) \approx -0.146,$$

so a 14.6% drop in investment prices hits the singularity. In comparison, in Greenwood et al. (2000) deviations of the price of investment from trend stay comfortably at less than 10%, so the problem remains bounded for one-sector models.

In a multi-sector model, however, a TFP shock to a single sector does not stay confined. Instead, it propagates through the production network and affects investment prices economy-wide. Moreover, sectors differ in their depreciation rates and hence in their distance to the singularity, and shocks differ in size. Whether a shock to sector i pushes the economy close to a singularity depends on three factors: (i) how the shock propagates to investment prices across the network, (ii) the distance of each affected sector to its singularity, and (iii) how these sectoral effects aggregate. We now use the partial equilibrium structure to build a singularity exposure measure that combines these three factors.

(i) *Network propagation.* A TFP shock to sector i affects output prices through direct and indirect input-output linkages captured by the Leontief inverse $L = (I - A)^{-1}$. These price changes then feed into investment goods prices through the capital flow matrix B. Combining both, a shock to sector i changes sector j's log investment price by approximately ψ_{ji}, where

$$\Psi = BL, \quad \psi_{ji} = \sum_{k=1}^{n} b_{jk}\, \ell_{ki}.$$

(ii) *Distance to singularity.* From the partial equilibrium analysis, the singularity for sector j lies at distance $|\varepsilon_j^*| = |\log(\beta(1-\delta_j))|$ from steady state in log investment price space. A shock of size ε_i to sector i therefore covers a fraction

$$\frac{|\psi_{ji}\, \varepsilon_i|}{|\log(\beta(1-\delta_j))|}$$

of the distance to sector j's singularity. The closer this ratio is to one, the worse the perturbation approximation for sector j's capital demand.

(iii) *Aggregation.* Since the elasticities in Figure 3 are for aggregate variables, the aggregate approximation error from a shock to sector i depends on the size of the affected sectors. Weighting by Domar weights w_j and scaling by the standard deviation of sector i's TFP, $\sigma(\ln z_i)$, we define the *singularity exposure index* of sector i as:[5]

$$SE_i = \sigma(\ln z_i) \sum_{j=1}^{n} w_j \frac{\psi_{ji}}{|\log(\beta(1-\delta_j))|}$$

where w_j are Domar weights. Singularity exposure is high when sector i has volatile

[5] In our calibration, sectoral TFP shocks have a full variance-covariance matrix Σ^z, so a shock to sector i also moves correlated sectors. However, the elasticity exercise in Figure 3 perturbs one sector at a time, so $\sigma(\ln z_i)$ is the relevant scale for the impulse in that comparison.

TFP ($\sigma(\ln z_i)$ large), strongly influences investment prices (ψ_{ji} large) for sectors near their singularity ($|\log(\beta(1-\delta_j))|$ small), and does so for large sectors (w_j large).

The main prediction is that sectors with high singularity exposure index should exhibit larger disagreements between the deep learning and perturbation solutions. To test this, we simulate the model under each solution method, shocking only sector i, and measure the gap as the absolute difference in the implied standard deviation of aggregate labor and investment. This gap captures how much the perturbation method over- or understates the aggregate volatility attributable to each sector.

Table 4: Correlations of sectoral metrics with the volatility gap between the deep learning and perturbation solutions

	DL vs. First-order		DL vs. Third-order	
Metrics	Labor	Invest.	Labor	Invest.
w_i	0.301	0.253	0.303	0.227
δ_i	0.533	0.494	0.465	0.348
α_i^k	−0.076	−0.013	−0.079	0.035
α_i^h	−0.026	−0.037	0.016	0.006
$\sigma(\ln z_i)$	0.383	0.430	0.381	0.481
SE_i	0.785	0.718	0.780	0.699

Table 4 reports the correlation of the volatility gap with SE_i and with individual sectoral characteristics. Among individual factors, the depreciation rate δ_i shows the strongest correlation, consistent with the resource constraint channel: high-depreciation sectors dedicate a larger and more variable share of output to investment, amplifying perturbation errors. However, the correlation with δ_i alone is moderate (0.35–0.53). Domar weights are also positively correlated (0.23–0.30), as larger sectors contribute more to aggregate volatility and thus to the aggregate gap. Capital and labor shares have limited explanatory power on their own. TFP volatility $\sigma(\ln z_i)$ is positively correlated with the gap (0.38–0.48), since more volatile sectors generate larger shocks, but the correlation is modest. The singularity exposure index SE_i achieves substantially higher correlations (0.70–0.79) across all specifications, confirming that the interaction of network structure, singularity proximity, and shock volatility is the primary driver of perturbation failure.

Finally, Table 5 reports results from a multiple regression of the volatility gap on all sectoral characteristics and SE_i jointly. The singularity exposure index is the only consistently significant predictor ($p < 0.05$) across all four specifications, while no individual component

is consistently significant. This confirms that the interaction of network structure and singularity proximity captured by SE_i contains information beyond what individual sectoral characteristics can explain.

Table 5: Multiple regression of the volatility gap on sectoral metrics

	DL vs. First-order		DL vs. Third-order	
	Labor	Invest.	Labor	Invest.
w_i	0.040^{**}	0.203	0.037^{**}	0.145
	(0.018)	(0.120)	(0.018)	(0.115)
δ_i	0.029^{*}	0.185^{*}	0.018	0.068
	(0.014)	(0.095)	(0.014)	(0.092)
α_i^k	-0.004	-0.012	-0.004	-0.005
	(0.004)	(0.025)	(0.004)	(0.024)
α_i^h	0.000	0.003	0.001	0.006
	(0.003)	(0.020)	(0.003)	(0.019)
$\sigma(\ln z_i)$	0.041	0.345	0.036	0.336
	(0.031)	(0.208)	(0.031)	(0.200)
SE_i	0.050^{***}	0.224^{**}	0.052^{***}	0.232^{**}
	(0.016)	(0.106)	(0.016)	(0.102)
R^2	0.703	0.601	0.671	0.541
Adj. R^2	0.644	0.521	0.606	0.449

$^{*}p < 0.1$, $^{**}p < 0.05$, $^{***}p < 0.01$

7 Conclusion

This paper developed a scalable deep learning framework to globally solve dynamic general equilibrium models with production and investment networks, exploiting the underlying symmetry in the economic environment to overcome the curse of dimensionality. Applied to a calibrated 37-sector model of the U.S. economy, we showed that global nonlinearities substantially alter equilibrium dynamics relative to perturbation methods. Our main result is that the deep learning solution reproduces the slow recovery of output and investment from the Great Recession observed in the data, while both perturbation solutions counterfactually predict a rapid rebound. The mechanism is a bottleneck in investment hubs: depleted capital depresses labor demand, which limits investment goods production, preventing capital from recovering. Perturbation methods miss this because they extrapolate the surge in capital demand to violate the nonlinear resource constraints.

We traced this failure of perturbation methods to singularities in the Euler equation for capital, where the user cost of capital approaches zero and capital demand diverges. In a network economy, a shock to a single sector propagates through input-output and investment linkages, pushing multiple sectors toward their singularities simultaneously. We proposed a novel singularity exposure index, computable directly from calibration parameters, that quantifies this vulnerability and showed that it is the dominant predictor of the gap between the deep learning and perturbation solutions.

References

Acemoglu, D., V. M. Carvalho, A. Ozdaglar, and A. Tahbaz-Salehi (2012). The network origins of aggregate fluctuations. *Econometrica 80*(5), 1977–2016.

Andreasen, M. M., J. Fernández-Villaverde, and J. F. Rubio-Ramírez (2018). The pruned state-space system for non-linear DSGE models: Theory and empirical applications. *The Review of Economic Studies 85*(1), 1–49.

Azinovic, M., L. Gaegauf, and S. Scheidegger (2022). Deep equilibrium nets. *International Economic Review 63*(4), 1471–1525.

Baqaee, D. R. and E. Farhi (2019). The macroeconomic impact of microeconomic shocks: Beyond hulten's theorem. *Econometrica 87*, 1155–1203.

Belkin, M., D. Hsu, S. Ma, and S. Mandal (2019). Reconciling modern machine-learning practice and the classical bias-variance trade-off. *Proceedings of the National Academy of Sciences 116*(32), 15849–15854.

Bellman, R. (1957). *Dynamic Programming*. Princeton University Press.

Carvalho, V. M., M. Covarrubias, and G. Nuño (2025). Planning against disasters in dynamic production networks. Working paper.

Cybenko, G. (1989). Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals, and Systems 2*(4), 303–314.

Ebrahimi Kahou, M., J. Fernández-Villaverde, S. Gomez-Cardona, J. Perla, and J. Rosa (2024). Spooky boundaries at a distance: Inductive bias, dynamic models, and behavioral macro. Working Paper 32850, National Bureau of Economic Research.

Ebrahimi Kahou, M., J. Fernández-Villaverde, J. Perla, and A. Sood (2021). Exploiting symmetry in high-dimensional dynamic programming. NBER Working Paper 28981, National Bureau of Economic Research.

Fernández-Villaverde, J., G. Nuño, and J. Perla (2024). Taming the curse of dimensionality: Quantitative economics with deep learning. Working Paper 33117, National Bureau of Economic Research.

Fernández-Villaverde, J., S. Hurtado, and G. Nuño (2023). Financial frictions and the wealth distribution. *Econometrica 91*, 869–901.

Foerster, A. T., P.-D. G. Sarte, and M. W. Watson (2011). Sectoral versus aggregate shocks: A structural factor analysis of industrial production. *Journal of Political Economy 119*(1), 1–38.

Greenwood, J., Z. Hercowitz, and P. Krusell (2000). The role of investment-specific technological change in the business cycle. *European economic review 44*(1), 91–115.

Hansen, G. D. (1985). Indivisible labor and the business cycle. *Journal of Monetary Economics 16*(3), 309–327.

He, K., X. Zhang, S. Ren, and J. Sun (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778.

Hornik, K. (1991). Approximation capabilities of multilayer feedforward networks. *Neural Networks 4*(2), 251–257.

Kase, H., L. Melosi, and M. Rottner (2025). Estimating nonlinear heterogeneous agent models with neural networks. Working paper.

Kim, J., S. Kim, E. Schaumburg, and C. A. Sims (2008). Calculating and using second-order accurate solutions of discrete time dynamic equilibrium models. *Journal of Economic Dynamics and Control 32*(11), 3397–3414.

Long, J. B. and C. I. Plosser (1983). Real business cycles. *Journal of political Economy 91*(1), 39–69.

Maliar, L., S. Maliar, and P. Winant (2021). Deep learning for solving dynamic economic models. *Journal of Monetary Economics 122*, 76–101.

McCloskey, M. and N. J. Cohen (1989). Catastrophic interference in connectionist networks: The sequential learning problem. *The Psychology of Learning and Motivation 24*, 109–165.

vom Lehn, C. and T. Winberry (2022). The investment network, sectoral comovement, and the changing u.s. business cycle. *Quarterly Journal of Economics 137*(1), 387–433.

Zaheer, M., S. Kottur, S. Ravanbakhsh, B. Poczos, R. R. Salakhutdinov, and A. J. Smola (2017). Deep sets. In *Advances in Neural Information Processing Systems*, pp. 3391–3401.

Appendix

A Equilibrium Conditions

$$\mu_{i,t} = \beta \mathbb{E}_t \left[(C_{t+1})^{-\sigma} p_{i,t+1} \alpha_i^k \frac{y_{i,t+1}}{k_{i,t+1}} + \mu_{i,t+1}(1-\delta_i) \right], \quad i = 1, \ldots, n$$

$$C_t^{-\sigma} w_t = 1$$

$$p_{i,t} \alpha_i^h y_{i,t} = w_t h_{i,t}, \quad i = 1, \ldots, n$$

$$p_{i,t} a_{ij} y_{i,t} = p_{j,t} m_{ij,t}, \quad i, j = 1, \ldots, n$$

$$\mu_{i,t} b_{ij} x_{i,t} = C_t^{-\sigma} p_{j,t} q_{ij,t}, \quad i, j = 1, \ldots, n$$

$$y_{i,t} = c_{i,t} + \sum_{j=1}^{n} m_{ji,t} + \sum_{j=1}^{n} q_{ji,t}, \quad i = 1, \ldots, n$$

$$k_{i,t+1} = (1-\delta_i) k_{i,t} + x_{i,t}, \quad i = 1, \ldots, n$$

$$y_{i,t} = e^{z_{i,t}} k_{i,t}^{\alpha_i^k} h_{i,t}^{\alpha_i^h} \prod_{j=1}^{n} m_{ij,t}^{a_{ij}}, \quad i = 1, \ldots, n$$

$$x_{i,t} = \prod_{j=1}^{n} q_{ij,t}^{b_{ij}}, \quad i = 1, \ldots, n$$

$$p_{i,t} c_{i,t} = \omega_i C_t, \quad i = 1, \ldots, n$$

$$C_t = \prod_{i=1}^{n} (c_{i,t})^{\omega_i}$$

B Steady States

In the steady state,

$$\begin{aligned}
&\mu_i = \beta\left\{(C)^{-\sigma}p_i\alpha_i^k\frac{y_i}{k_i} + \mu_i(1-\delta_i)\right\}, \quad i = 1,\ldots,n\\
&C^{-\sigma}w = 1\\
&p_i\alpha_i^h y_i = wh_i, \quad i = 1,\ldots,n\\
&p_i a_{ij} y_i = p_j m_{ij}, \quad i,j = 1,\ldots,n\\
&\mu_i b_{ij} x_i = C^{-\sigma}p_j q_{ij}, \quad i,j = 1,\ldots,n\\
&y_i = c_i + \sum_{j=1}^{n} m_{ji} + \sum_{j=1}^{n} q_{ji}, \quad i = 1,\ldots,n\\
&\delta_i k_i = x_i, \quad i = 1,\ldots,n\\
&y_i = k_i^{\alpha_i^k} h_i^{\alpha_i^h}\prod_{j=1}^{n} m_{ij}^{a_{ij}}, \quad i = 1,\ldots,n\\
&x_i = \prod_{j=1}^{n} q_{ij}^{b_{ij}}, \quad i = 1,\ldots,n\\
&p_i c_i = \omega_i C, \quad i = 1,\ldots,n\\
&C = \prod_{i=1}^{n}(c_i)^{\omega_i}
\end{aligned}$$

From Euler equation

$$\begin{aligned}
\mu_i &= \beta\left[(C)^{-\sigma}p_i\alpha_i^k\frac{y_i}{k_i} + \mu_i(1-\delta_i)\right]\\
\frac{y_i}{k_i} &= \frac{\mu_i\left[\beta^{-1}+\delta-1\right]}{(C)^{-\sigma}p_i\alpha_i^k}
\end{aligned}$$

so take logs we have

$$\ln y_i - \ln k_i = \ln\mu_i - \ln p_i + \sigma\ln C + \ln\left[\frac{\beta^{-1}+\delta_i-1}{\alpha_i^k}\right]$$

and in a vector form

$$\ln\boldsymbol{k} = \ln\boldsymbol{y} - \ln\boldsymbol{\mu} + \ln\boldsymbol{p} - \sigma\ln C\mathbf{1} - \ln\left[(\beta^{-1}-1)\mathbf{1}+\boldsymbol{\delta}\right] + \ln\boldsymbol{\alpha}^k \tag{10}$$

where $\boldsymbol{k} = [k_1,\ldots,k_n]', \boldsymbol{y} = [y_1,\ldots,y_n]', \boldsymbol{\mu} = [\mu_1,\ldots,\mu_n]', \boldsymbol{p} = [p_1,\ldots,p_n]', \boldsymbol{\alpha}^k = [\alpha_1^k,\ldots,\alpha_n^k]'$ and $\mathbf{1}$ is a $n\times 1$ vector of ones.

Evaluating investment input demand condition at the steady state and taking logs and rearranging, we have

$$\ln q_{ij} = \ln \mu_i + \ln b_{ij} + \ln x_i + \sigma \ln C - \ln p_j.$$

Define a $(\sum_{i=1}^{n} n_{q_i} \times 1)$ vector $\boldsymbol{q}$ that collects all non-zero elements of q_{ij}:

$$\boldsymbol{q} \equiv [q_{11}, \ldots, q_{1n}, \ldots, q_{n1}, \ldots, q_{nn}]', \tag{11}$$

where n_{q_i} is the number of non-zero elements in $[q_{i1}, \ldots, q_{in}]'$. Further define the following matrices:

$$\underset{\sum_{i=1}^{n} n_{q_i} \times n}{M_\mu} = M_x \equiv \begin{bmatrix} \underset{n_{q_1} \times 1}{\mathbf{1}} & & \mathbf{0} \\ & \ddots & \\ \mathbf{0} & & \underset{n_{q_n} \times 1}{\mathbf{1}} \end{bmatrix},$$

where $\mathbf{1}$ denotes a vector of ones. In addition, M_p is a $(\sum_{i=1}^{n} n_{q_i} \times n)$ matrix where we remove rows corresponding to zero elements in $vec(B')$ from a matrix $\underset{n \times 1}{\mathbf{1}} \otimes \underset{n \times n}{\boldsymbol{I}}$, where $\boldsymbol{I}$ is an identity matrix. Then the optimality condition can be written in a vector form as

$$\ln \boldsymbol{q} = M_\mu \ln \boldsymbol{\mu} + \ln \widehat{B} + M_x \ln \boldsymbol{x} + \sigma \ln C \widetilde{\mathbf{1}} - M_p \ln \boldsymbol{p}, \tag{12}$$

where $\boldsymbol{\mu} \equiv [\mu_1, \ldots, \mu_n]'$, $\boldsymbol{x} \equiv [x_1, \ldots, x_n]'$, $\boldsymbol{p} \equiv [p_1, \ldots, p_n]'$, and $\widehat{B}$ is a vector of b_{ij}s constructed by removing zero elements from $vec(B')$. $\widetilde{\mathbf{1}}$ is a $\sum_{i=1}^{n} n_{q_i} \times 1$ vector of ones.

Evaluating (4) at the steady state and taking logs, we have

$$\ln x_i = \sum_{j=1}^{n} b_{ij} \ln q_{ij},$$

where in a vector form can be rewritten as

$$\ln \boldsymbol{x} = \widetilde{B} \ln \boldsymbol{q}, \tag{13}$$

where $\widetilde{B}$ is a matrix constructed by removing zero elements from $vec(B')$ such that

$$\underset{n\times\sum_{i=1}^{n} n_{q_i}}{\widetilde{B}} = \begin{bmatrix} b_{11},\ldots,b_{1n} & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & b_{21},\ldots,b_{2n} & & \vdots \\ \vdots & & \ddots & \mathbf{0} \\ \mathbf{0} & \cdots & \mathbf{0} & b_{n1},\ldots,b_{nn} \end{bmatrix}.$$

Substitute (12) into (13) and obtain

$$\ln \boldsymbol{x} = \underbrace{\widetilde{B}M_{\mu}}_{=\boldsymbol{I}} \ln \boldsymbol{\mu} + \widetilde{B}\ln\widehat{B} + \underbrace{\widetilde{B}M_{x}}_{=\boldsymbol{I}} \ln \boldsymbol{x} + \widetilde{B}\widetilde{\mathbf{1}}\sigma \ln C - \underbrace{\widetilde{B}M_{p}}_{=B} \ln \boldsymbol{p},$$

so we have

$$\ln \boldsymbol{\mu} = B\ln \boldsymbol{p} - \widetilde{B}\ln\widehat{B} - \widetilde{B}\widetilde{\mathbf{1}}\sigma \ln C. \tag{14}$$

Substitute into

$$\ln \boldsymbol{k} = \ln \boldsymbol{y} - \ln \boldsymbol{\mu} + \ln \boldsymbol{p} - \sigma \ln C\mathbf{1} - \ln\left[(\beta^{-1}-1)\mathbf{1} + \boldsymbol{\delta}\right] + \ln \boldsymbol{\alpha}^{k}$$
$$\ln \boldsymbol{k} = \ln \boldsymbol{y} + [\boldsymbol{I} - B]\ln \boldsymbol{p} + \widetilde{B}\ln\widehat{B} + \left[\widetilde{B}\widetilde{\mathbf{1}} - \mathbf{1}\right]\sigma \ln C - \ln\left[(\beta^{-1}-1)\mathbf{1} + \boldsymbol{\delta}\right] + \ln \boldsymbol{\alpha}^{k}$$

Consider a vector of non-zero sectoral steady-state consumption $\boldsymbol{c} = [c_1,\ldots,c_n]'$. Then we have

$$\ln C = \widetilde{\boldsymbol{\omega}}' \ln \boldsymbol{c},$$

where $\widetilde{\boldsymbol{\omega}} = [\omega_1,\ldots,\omega_n]'$ is a vector where we removed zero elements from $\boldsymbol{\omega}$. Using the consumption demand function, we have

$$M_{cp}\ln \boldsymbol{p} = \ln \widetilde{\boldsymbol{\omega}} + [\mathbf{1}\widetilde{\boldsymbol{\omega}}' - \boldsymbol{I}]\ln \boldsymbol{c},$$

where $\boldsymbol{p} = [p_1,\ldots,p_n]'$ and M_{cp} is a matrix where we remove from I, rows corresponding to zero elements in $\boldsymbol{\omega}$.

Evaluating the input demand at the steady state, the intermediate input demand can be expressed as

$$\ln m_{ij} = \ln p_i - \ln p_j + \ln a_{ij} + \ln y_i.$$

Define a $(\sum_{i=1}^{n} n_{m_i} \times 1)$ vector m that collects all non-zero elements of m_{ij}:

$$\boldsymbol{m} \equiv [m_{11},\ldots,m_{1n},\ldots,m_{n1},\ldots,m_{nn}]', \tag{15}$$

where n_{m_i} is the number of non-zero elements in $[m_{i1}, \ldots, m_{in}]'$. Then (15) can be expressed in a vector form as

$$\ln \boldsymbol{m} = M_m \ln \boldsymbol{p} + M_y \ln \boldsymbol{y} + \ln \widehat{A} \tag{16}$$

where $M_m = M_{m_1} - M_{m_2}$ and define

$$\underset{\sum_{i=1}^{n} n_{m_i} \times n}{M_{m_1}} = \underset{\sum_{i=1}^{n} n_{m_i} \times n}{M_y} \equiv \begin{bmatrix} \underset{n_{m_1} \times 1}{\mathbf{1}} & & \mathbf{0} \\ & \ddots & \\ \mathbf{0} & & \underset{n_{m_n} \times 1}{\mathbf{1}} \end{bmatrix},$$

and M_{m_2} is a $(\sum_{i=1}^{n} n_{m_i} \times n)$ matrix where we remove rows corresponding to zero elements in $vec(A')$ from a matrix $\underset{n \times 1}{\mathbf{1}} \otimes \underset{n \times n}{\mathbf{I}}$. Finally, $\widehat{A}$ is a vector of a_{ij}s constructed by removing zero elements from $vec(A')$. From the household labor supply condition, $w = C^{\sigma}$. From the labor demand condition,

$$\ln \boldsymbol{p} + \ln \boldsymbol{\alpha}^h + \ln \boldsymbol{y} = \sigma \mathbf{1} \widetilde{\boldsymbol{\omega}}' \ln \boldsymbol{c} + \ln \boldsymbol{h}$$

where $\boldsymbol{h} = [h_1, \ldots, h_n]'$, $\boldsymbol{\alpha}^h = [\alpha_1^h, \ldots, \alpha_n^h]'$.

From the production function

$$\ln \boldsymbol{y} = diag(\boldsymbol{\alpha}^k) \ln \boldsymbol{k} + diag(\boldsymbol{\alpha}^h) \ln \boldsymbol{h} + \widetilde{A} \ln \boldsymbol{m}$$

and $\widetilde{A}$ is a matrix constructed by removing zero elements from $vec(A')$ such that

$$\underset{n \times \sum_{i=1}^{n} n_{m_i}}{\widetilde{A}} = \begin{bmatrix} a_{11}, \ldots, a_{1n} & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & a_{21}, \ldots, a_{2n} & & \vdots \\ \vdots & & \ddots & \mathbf{0} \\ \mathbf{0} & \cdots & \mathbf{0} & a_{n1}, \ldots, a_{nn} \end{bmatrix}.$$

Substitute in $\ln \boldsymbol{k}$, $\ln \boldsymbol{h}$ and $\ln \boldsymbol{m}$:

$$\begin{aligned} &\ln \boldsymbol{y} \\ &= diag(\boldsymbol{\alpha}^k) \left[[\boldsymbol{I} - B] \ln \boldsymbol{p} + \widetilde{B} \ln \widehat{B} + \left[\widetilde{B} \widetilde{\mathbf{1}} - \mathbf{1} \right] \sigma \widetilde{\boldsymbol{\omega}}' \ln \boldsymbol{c} - \ln \left[(\beta^{-1} - 1) \mathbf{1} + \boldsymbol{\delta} \right] + \ln \boldsymbol{\alpha}^k \right] \\ &+ diag(\boldsymbol{\alpha}^h) \left[\ln \boldsymbol{p} + \ln \boldsymbol{\alpha}^h - \sigma \mathbf{1} \widetilde{\boldsymbol{\omega}}' \ln \boldsymbol{c} \right] \\ &+ \widetilde{A} \left[M_m \ln \boldsymbol{p} + \ln \widehat{A} \right] \end{aligned}$$

so we have

$$\ln \boldsymbol{p} = -\left[diag(\boldsymbol{\alpha}^k)\left[\boldsymbol{I} - B\right] + diag(\boldsymbol{\alpha}^h) + \widetilde{A}M_m\right]^{-1}$$
$$\times \left[diag(\boldsymbol{\alpha}^k)\left[\widetilde{B}\ln\widehat{B} + \left[\widetilde{B}\widetilde{\mathbf{1}} - \mathbf{1}\right]\sigma\widetilde{\boldsymbol{\omega}}'\ln \boldsymbol{c} - \ln\left[(\beta^{-1}-1)\mathbf{1} + \boldsymbol{\delta}\right] + \ln\boldsymbol{\alpha}^k\right] + diag(\boldsymbol{\alpha}^h)\left[\ln\boldsymbol{\alpha}^h - \sigma\mathbf{1}\widetilde{\boldsymbol{\omega}}'\ln \boldsymbol{c}\right.$$

Pre-multiply both sides by M_{cp}:

$$\ln\widetilde{\boldsymbol{\omega}} + [\mathbf{1}\widetilde{\boldsymbol{\omega}}' - \boldsymbol{I}]\ln \boldsymbol{c}$$
$$= -\ln\widetilde{\boldsymbol{\omega}} - M_{cp}\left[diag(\boldsymbol{\alpha}^k)\left[\boldsymbol{I} - B\right] + diag(\boldsymbol{\alpha}^h) + \widetilde{A}M_m\right]^{-1}$$
$$\times \left[diag(\boldsymbol{\alpha}^k)\left[\widetilde{B}\ln\widehat{B} - \ln\left[(\beta^{-1}-1)\mathbf{1} + \boldsymbol{\delta}\right] + \ln\boldsymbol{\alpha}^k\right] + diag(\boldsymbol{\alpha}^h)\ln\boldsymbol{\alpha}^h + \widetilde{A}\ln\widehat{A}\right]$$
$$\ln \boldsymbol{c}$$
$$= -\left[\mathbf{1}\widetilde{\boldsymbol{\omega}}' - \boldsymbol{I} + M_{cp}\left[diag(\boldsymbol{\alpha}^k)\left[\boldsymbol{I} - B\right] + diag(\boldsymbol{\alpha}^h) + \widetilde{A}M_m\right]^{-1}\left[diag(\boldsymbol{\alpha}^k)\left[\widetilde{B}\widetilde{\mathbf{1}} - \mathbf{1}\right] - diag(\boldsymbol{\alpha}^h)\mathbf{1}\right]\sigma\widetilde{\boldsymbol{\omega}}'\right]^{-1}$$
$$\times \left[\ln\widetilde{\boldsymbol{\omega}} + M_{cp}\left[diag(\boldsymbol{\alpha}^k)\left[\boldsymbol{I} - B\right] + diag(\boldsymbol{\alpha}^h) + \widetilde{A}M_m\right]^{-1}\left[diag(\boldsymbol{\alpha}^k)\left[\widetilde{B}\ln\widehat{B} - \ln\left[(\beta^{-1}-1)\mathbf{1} + \boldsymbol{\delta}\right] + \ln\boldsymbol{\alpha}^k\right]\right.\right.$$

We can calculate $\ln\boldsymbol{\mu}$ from (14):

$$\ln\boldsymbol{\mu} = B\ln\boldsymbol{p} - \widetilde{B}\ln\widehat{B} - \widetilde{B}\widetilde{\mathbf{1}}\sigma\ln C.$$

Resource constraint

$$\boldsymbol{c} + M_{my}\boldsymbol{y} + M_{qx}\boldsymbol{x} = \boldsymbol{y}$$

where

$$M_{my} = \begin{bmatrix} \frac{p_1}{p_1}a_{11} & \dots & \frac{p_n}{p_1}a_{n1} \\ \vdots & \ddots & \vdots \\ \frac{p_1}{p_n}a_{1n} & \dots & \frac{p_n}{p_n}a_{nn} \end{bmatrix}, \quad M_{qx} = \begin{bmatrix} C^\sigma\frac{\mu_1}{p_1}b_{11} & \dots & C^\sigma\frac{\mu_n}{p_1}b_{n1} \\ \vdots & \ddots & \vdots \\ C^\sigma\frac{\mu_1}{p_n}b_{1n} & \dots & C^\sigma\frac{\mu_n}{p_n}b_{nn} \end{bmatrix}.$$

Take an exponential on (10):

$$k_i = \underbrace{\exp\left(-\ln\mu_i + \ln p_i - \sigma\ln C + \ln\left[\frac{\alpha_i^k}{\beta^{-1} + \delta_i - 1}\right]\right)}_{\equiv G_i} y_i,$$

and using (3),

$$\boldsymbol{x} = diag(\boldsymbol{\delta})\boldsymbol{k} = diag(\boldsymbol{\delta})M_k\boldsymbol{y},$$

where $M_k = diag\left([G_1, \ldots, G_n]'\right)$. Hence

$$\begin{aligned}
&\boldsymbol{c} + M_{my}\boldsymbol{y} + diag(\boldsymbol{\delta})M_{qx}M_k\boldsymbol{y} = \boldsymbol{y} \\
&\boldsymbol{y} = (\boldsymbol{I} - M_{my} - diag(\boldsymbol{\delta})M_{qx}M_k)^{-1}\,\boldsymbol{c}
\end{aligned}$$

Derive $\boldsymbol{k}$ using

$$\ln \boldsymbol{k} = \ln \boldsymbol{y} - \ln \boldsymbol{\mu} + \ln \boldsymbol{p} - \sigma \ln C\mathbf{1} - \ln\left[(\beta^{-1} - 1)\mathbf{1} + \boldsymbol{\delta}\right] + \ln \boldsymbol{\alpha}^k$$

Derive $\boldsymbol{m}$ using

$$\ln \boldsymbol{m} = M_m \ln \boldsymbol{p} + M_y \ln \boldsymbol{y} + \ln \widehat{A}$$

Then from the production function

$$\ln \boldsymbol{h} = (diag(\boldsymbol{\alpha}^h))^{-1}\left[\ln \boldsymbol{y} - diag(\boldsymbol{\alpha}^k)\ln \boldsymbol{k} - \widetilde{A}\ln \boldsymbol{m}\right]$$

Aggregate capital stock

$$K = \sum_{i=1}^{n} k_i = \mathbf{1}'\boldsymbol{k}$$

Aggregate hours worked

$$H = \sum_{i=1}^{n} h_i = \mathbf{1}'\boldsymbol{h}$$

Aggregate investment

$$X = \sum_{i=1}^{n} x_i = \mathbf{1}'\boldsymbol{x}$$

Aggregate output

$$Y = C + X$$

Value function for sector i

$$\begin{aligned}
V_i &= C^{-\sigma}\left[p_i y_i - \sum_{j=1}^{n} p_j m_{ij} - \sum_{j=1}^{n} p_j q_{ij} - w h_i\right] + \beta V_i \\
\boldsymbol{V} &= (1-\beta)^{-1} C^{-\sigma}\left[diag(\boldsymbol{p})\boldsymbol{y} - (M+Q)\boldsymbol{p} - w\boldsymbol{h}\right]
\end{aligned}$$

where M is the $n \times n$ matrix of intermediate inputs with elements m_{ij}, and Q is the $n \times n$

matrix of investment inputs with elements q_{ij}. We can also define the steady-state profit:

$$\boldsymbol{\Pi} = diag(\boldsymbol{p})\boldsymbol{y} - (M + Q)\boldsymbol{p} - w\boldsymbol{h}$$

C Computing Equilibrium Allocations from the Deep Learning Solution

Given the deep learning approximations $\hat{c}_i, \hat{x}_i, \hat{V}_i$ (log-deviations from steady state), the remaining equilibrium allocations $\hat{y}_i, \hat{h}_i, \hat{p}_i, \hat{\mu}_i, \hat{m}_{ij}, \hat{q}_{ij}$ can be computed exactly from the equilibrium conditions. We use the matrices defined in the previous section and denote log-deviations from the steady state with hats.

Prices. From the Cobb-Douglas consumption aggregator (1), aggregate consumption in log-deviations is

$$\hat{C} = \sum_{i=1}^{n} \omega_i \hat{c}_i.$$

From the consumption demand condition $p_i c_i = \omega_i C$, taking logs and subtracting the steady-state counterpart, we obtain sectoral prices:

$$\hat{p}_i = \hat{C} - \hat{c}_i, \quad i = 1, \ldots, n.$$

Wages. From the labor supply condition $w = C^{\sigma}$:

$$\hat{w} = \sigma \hat{C}.$$

Shadow value of capital. From (14), taking deviations:

$$\hat{\boldsymbol{\mu}} = B\hat{\boldsymbol{p}} - \sigma \mathbf{1}\hat{C}.$$

Output. From the production function and labor demand condition, substituting in $\hat{\boldsymbol{h}}$ and $\hat{\boldsymbol{m}}$:

$$\begin{aligned}\hat{\boldsymbol{y}} = &\left[\boldsymbol{I} - diag(\boldsymbol{\alpha}^h) - \widetilde{A}M_y\right]^{-1} \\ &\times \left[\hat{\boldsymbol{z}} + diag(\boldsymbol{\alpha}^k)\hat{\boldsymbol{k}} + diag(\boldsymbol{\alpha}^h)\left[\hat{\boldsymbol{p}} - \mathbf{1}\hat{w}\right] + \widetilde{A}M_m\hat{\boldsymbol{p}}\right].\end{aligned}$$

Labor. From the labor demand condition:

$$\hat{\boldsymbol{h}} = \hat{\boldsymbol{p}} + \hat{\boldsymbol{y}} - \mathbf{1}\hat{w}.$$

Intermediate inputs. From the input demand condition:

$$\hat{\boldsymbol{m}} = M_m \hat{\boldsymbol{p}} + M_y \hat{\boldsymbol{y}}.$$

Investment inputs. From the investment input demand condition:

$$\hat{\boldsymbol{q}} = M_\mu \hat{\boldsymbol{\mu}} + M_x \hat{\boldsymbol{x}} + \sigma \widetilde{\mathbf{1}} \hat{C} - M_p \hat{\boldsymbol{p}}.$$

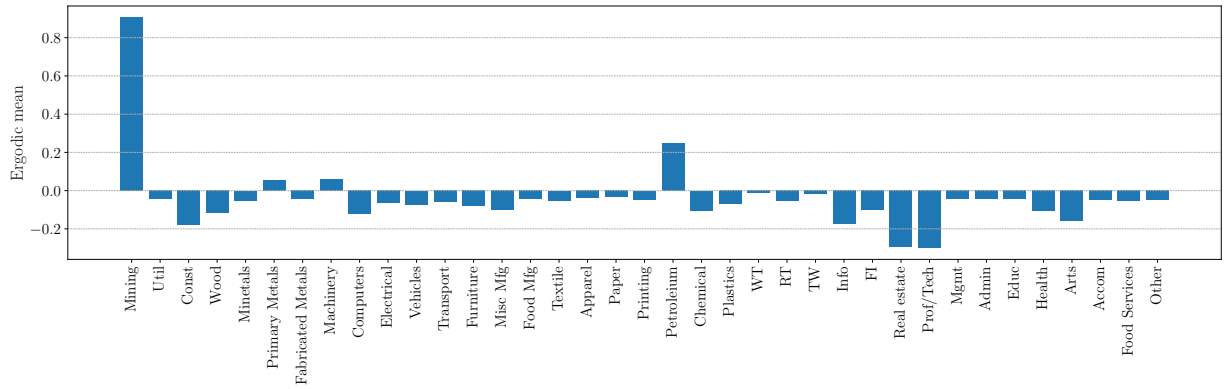
D List of Sectors

Table 6: List of sectors used in the analysis

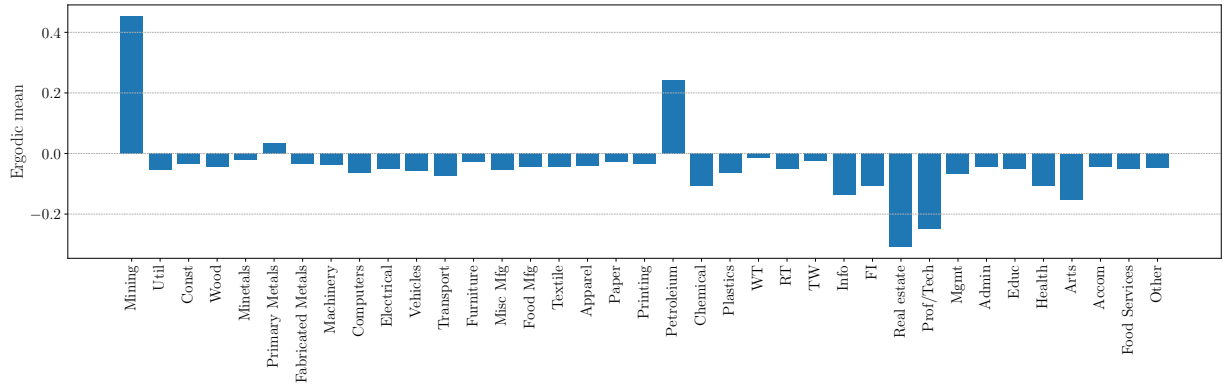
Mining	Petroleum and coal products
Utilities	Chemical products
Construction	Plastics and rubber products
Wood products	Wholesale trade
Nonmetallic mineral products	Retail trade
Primary metals	Transportation and warehousing
Fabricated metal products	Information
Machinery	Finance and insurance
Computer and electronic products	Real estate and rental and leasing services
Electrical equipment, appliances, and components	Professional and technical services
Motor vehicles, bodies and trailers, and parts	Management of companies and enterprises
Other transportation equipment	Administrative and waste management services
Furniture and related products	Educational services
Miscellaneous manufacturing	Health services
Food and beverage and tobacco products	Arts, entertainment, and recreation services
Textile mills and textile product mills	Accommodation services
Apparel and leather and allied products	Food services
Paper products	Other services, except government
Printing and related support activities	

E Additional Results

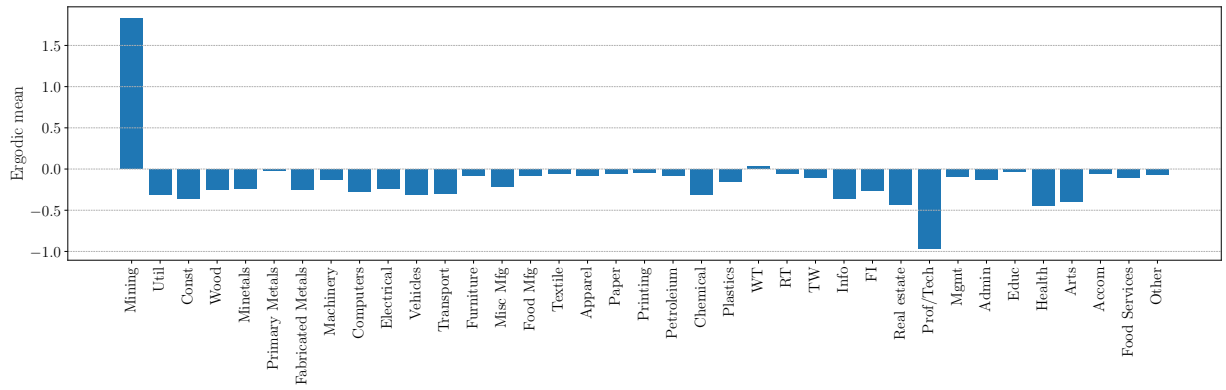
Figure 8: Sectoral distributions of ergodic means



a Sectoral gross output y_i

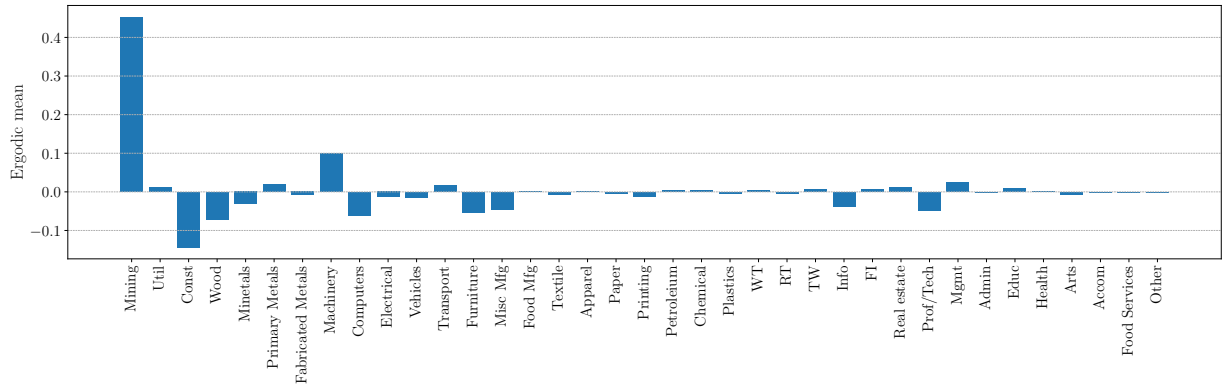


b Sectoral consumption c_i

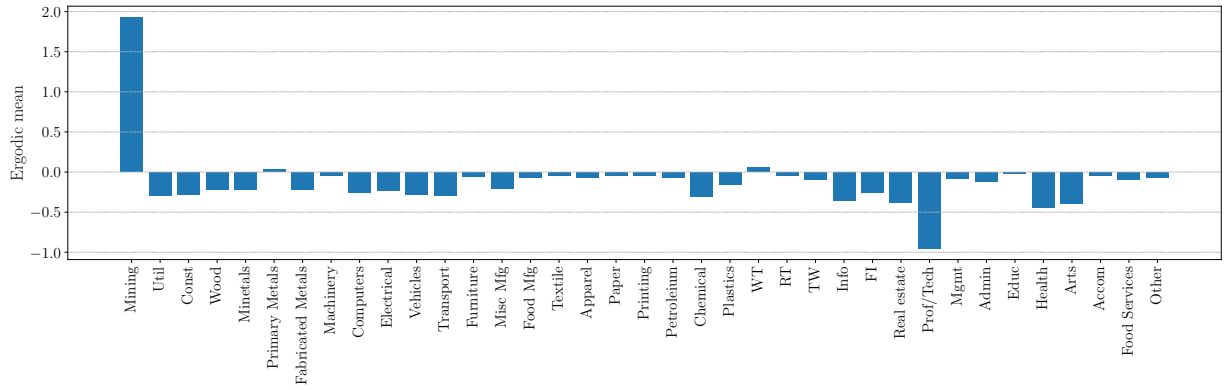


c Sectoral investment x_i

Figure 9: Sectoral distributions of ergodic means



a Sectoral labor h_i



b Sectoral capital k_i